Finding Novel Analogies

Diarmuid O'Donoghue, B.Sc. M.Sc. (UCC)

The thesis is submitted to University College Dublin for the degree of PhD in the Faculty of Science

(Submitted) October, 2004.

Department of Computer Science, UCD. Dublin, Ireland.

Supervised by Prof. M. T. Keane.

Table of Contents

4 A MODEL FOR IDENTIFYING CREATIVE ANALOGIES..... 97

5 TESTS OF STRUCTURAL RETRIEVAL 136

Summary

We describe a model for identifying novel analogies that supply inferences to a given target domain. The Kilaza model is a three-phase model of analogy encompassing the phases of retrieval, mapping and validation. Kilaza's retrieval model is based on the graph-structure of the target domain, overcoming the semantic restriction associated with other models. Kilaza uses a standard incremental model to identify the mapping and generate the inferences. Its validation model tests the validity of inferences before they are accepted. Validation is based primarily on argument restrictions enforced by the use of "functional attributes", which also support validation's adaptation component.

Extensive testing of Kilaza was based on two collections of domains, containing a total of 96 domain descriptions. The first collection was described by a small set of general relations, while the other collection used a diverse range of specific relations. Each domain in turn served as a target, while the other domains acted as sources. Kilaza's retrieval, mapping and validation models acted on the resulting analogies. The classifications given by Kilaza to the resulting inferences, were then examined by human raters to give an "absolute" measure of Kilaza's accuracy at validation and adaptation. Retrieval identified many mappings on the first collection, but wasn't successful on the collection that used a greater variety of relational predicates. In contrast, the validation model more accurately detected invalid inferences on the second collections.

Overall however, Kilaza was able to identify novel analogies without reference to the semantics of the given target problem.

I declare that this thesis is entirely my own work, carried out for University College Dublin, and has not been submitted for a degree to this or any other University and that the contents are original unless otherwise stated.

Signed

_____

Diarmuid O'Donoghue
October 17, 2004

Acknowledgements

I would firstly like to thank my supervisor, Prof. Mark Keane for his assistance and guidance over the years. I would also like to thank the Department of Computer Science, National University of Ireland, Maynooth, for their assistance during this project. I would especially like to thank my parents, family and friends for their support throughout this project. Finally, I would especially like to thank Amy for her support, love and understanding.

Chapter 1

Finding Novel Analogies

"All perception of truth is the detection of an analogy."

- - Henry David Thoreau, "The Journal of Henry D. Thoreau In Fourteen Volumes Bound as Two", Houghton Mifflin 1906, Republished by Dover Publications, 1962

1.1 Introduction

Analogy is a tool of the mind, which is used to construct new understanding. Its construction skills are used by many parts of the mind, and this makes the exact nature of analogy itself hard to discern. Its ubiquity means that the product of the tool and the tool itself, are both referred to as analogy. Analogy is such a useful tool that it is used explicitly by some of the greatest minds, as the following quote attests.

> *"…a young physicist engaged in the study of X-rays, often came to discuss with my husband the* analogies *one could expect to find between these rays, and their secondary rays, and the radiations of the radioactive bodies."*
>
> From "Pierre Curie" by Marie Curie (1923)

An analogy is a comparison between a poorly understood problem concept, and some more familiar but distinct set of information. The noted similarity

serves to generate useful predictions about that target. For example, we might draw an analogy between running in a marathon and sitting an examination. A lot of preparation is required to produce a good result, there is only a limited amount of time in which to prove yourself, and there is a lot of competition to be the best of all those involved in the challenge. We can see that there are two distinct topics involved in the analogy. One relates to the marathon event, while the other concerns the examination. The non-obvious similarity only comes to light when we closely examine the two domains of information.

The process of analogy allows us think of a partly understood problem as though it is just another example of some more familiar concept. We generally refer to the well-known experience as the *source* (or *base*) domain, while the less familiar problem is called the *target*. We refer to both as *domains* as they represent thematically related collections of information, and some or all of each domain may be used by the analogy.

Analogies have a wonderful ability to re-cycle familiar information and put it to a new and innovative use. Analogies are particularly impressive because they play an important role in so many other cognitive processes. Analogies are frequently used in teaching, re-applying familiar information to a new problem domain (Gick and Holyoak, 1980). So we might describe the structure of the atom by analogy to the more familiar solar system. Therefore, electrons orbit the nucleus just like the planets orbit the sun. It has been shown that by combining two suitably selected analogies, we can create a generic rule that is applicable across many related problems (Gick and Holyoak, 1983). Analogy also plays an important role in categorisation (Lakoff, 1987), and throughout the last century the rare and prehistoric looking "coelacanth" fish was assigned a variety of categorisations (Shelley, 1999). Each categorisation was based on analogy to different well-known fish. Polya (1957) describes the role of analogy in problem solving and *"the highest scientific achievements"*. Interestingly, analogy has also been shown to play an important role in creativity (Boden, 1992) and scientific discovery (Hoffman, 1995). Kekule's "invention" of aromatic chemistry was driven by an analogy between a chemical structure and a snake biting its own tail! In

fact, reasoning by analogy pervades our understanding of the world around us, and even of ourselves (Lakoff and Johnson, 1980). For example, we generally conceive of time as involving two containers, past and future, with the present nestled between them. Time flows from the future through the present and into the past.

1.1.1 Analogy and Metaphor

In this thesis, we make an important distinction between the terms *metaphor* and *analogy*, although they are sometimes used interchangeably. Typically, the term "metaphor" represents more artistic and illustrative uses of these comparisons. They generally add richness to the concepts being discussed, by comparison with a different but more familiar concept. Metaphors emphasise salient facts that are common to both source and target, while implicitly suppressing any information that differentiates them. Thus, they highlight new similarity where none was previously noticed. Metaphors perform a simple kind of learning by inserting *connections* between the base and target (Eskridge, 1994; Veale, 1995). These connections effectively reduce the semantic distance between the two domains.

> ***Metaphor*** *finds and highlights the similarity between two apparently dissimilar concepts.*

Analogies are often used in a more scientific context as they generate *new* facts about the target domain, rather than highlighting non-obvious similarity. Analogies use the identified similarity as a basis for transferring additional source information to the target, thereby generating *new* information within the target domain. Thus, the hearer must trust the speaker sufficiently to accept the additional information suggested by the comparison (Grice, 1975). This transfer of knowledge from the source to the target is a key characteristic of analogies, and one we shall be greatly concerned with in this thesis.

> ***Analogy*** *finds and then extends the similarity between domains, making new inferences about the problem domain.*

1.1.2 A Focus on Scientific Analogies

In this thesis we focus on scientific analogies because they allow us to concentrate on aspects of the analogy process that have received relatively little attention. Also, these analogies tend to be more straightforward than artistic uses of analogy. In scientific analogies, there is a once-off unidirectional interaction from source to target, and the target domain accepts the newly generated information. For example, consider the analogy "the heart is like a pump". This describes the heart's function, passing blood through the arteries to the body. Information about pumps can be applied to the heart, making its purpose much more understandable. (For a discussion on other scientific analogies see Hoffman, 1995).

In contrast, artistic uses of analogy are less well documented, involving perhaps multiple interactions between the two domains, and often depending upon arbitrary and unknown constraints. The meaning of such analogies is often open to interpretation, and the objective behind them can be very unclear. For example, it is difficult to state exactly what the following lines from Robert Burns' poem mean, or precisely why they are being said.

"*O, my luve's like a red, red rose,*
That's newly sprung in June".

Scientific analogies generally have a definite and identifiable objective - to help explain some scientific phenomenon. Therefore, we also know the context in which the scientific analogy is used. Especially in novel scientific analogies, the newly generated information is easily identified from the comparison. Clearly identifiable inferences also allow us to examine the processes that distinguish between acceptable and unacceptable inferences.

Good analogies generate useful predictions for the target domain, based on *extending* its similarity to the source domains. Extending the similarity is achieved by introducing information originating in the source domain into the target domain. So if the heart is like a pump, and we temporarily block a small tube connected to the pump, then removing the blockage should allow the flow to resume. It is this kind of reasoning that is the hallmark of

analogical reasoning. Strong analogies allow us to reason in the familiar source domain, while the implications of that reasoning process will be applicable to the target domain.

1.2 The Phases of Analogy

1.2.1 The Core of Analogical Mapping

The key to interpreting an analogy (or a metaphor) is to look first at the *structure* of the two domains (Gentner, 1983). Let us examine the structure of the earlier analogy "a marathon is like an examination" (see Figure 1.1). We can see that the two domains are identical in structure, and it is this structural *isomorphism* that is the key to interpreting a given analogy. So for any source-target pair, computationally we can treat these domains as graphs and identify the largest common sub-graph between them – the resulting collection of source:target pairs of information is referred to as "the mapping". A further aspect of analogy is brought to light if we overlay the two domains of Figure 1.1 and examine the overlaid relations. For example, we can see that "running-in" a marathon has a counterpart in "sitting" an examination in the other domain. At an abstract level, we can also observe a degree of semantic similarity between these pairs of counterpart relations.

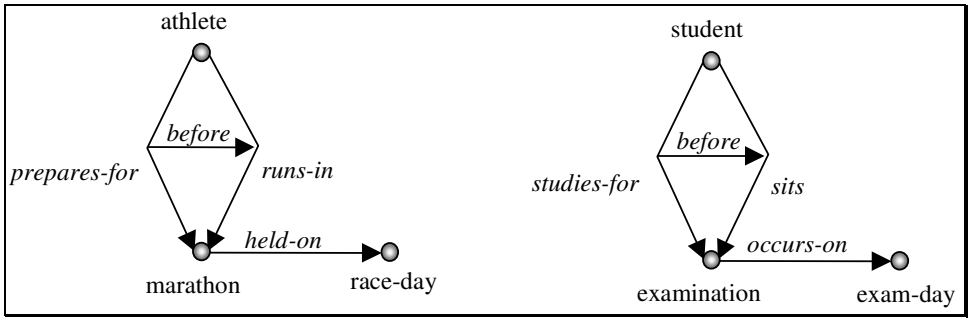

Figure 1. 1 - *Example analogy "a marathon is like an examination"*

A key property of analogical comparisons is *systematicity* (Gentner, 1983), and highlights that the two domains of an analogy use similar systems or relations in similar ways. Systematicity theory spawned much focused work on analogy (and metaphor), enabling computational modelling to verify and

compare various theories. Many researchers identify different phases in the analogy process, but there is a large degree of commonality between these meta-models of analogical reasoning. However, subtle distinctions in these meta-models can mask significant differences in the responsibilities assigned to the phases.

1.2.2 Phases of Analogy

We shall now examine a five-phase model of the analogy process (Keane, 1994), before briefly describing some alternative models. The first phase in the analogy process concerns *representing* each domain's knowledge as a collection of predicate calculus assertions. So for Gentner's description of Rutherford's analogy "an atom is like the solar-system" (1983) we may have: `attracts(sun,planet)` and `revolves-around(planet,sun)` as the predicate relations in the solar system domain. The second phase of analogy is *retrieval* and concerns identifying some stored domain to support inferences in the target domain. Retrieval occurs either when the source is provided in the environment, or is spontaneously retrieved from memory. So given the problem of describing the structure of an atom, we should identify the solar system domain as a candidate source. Third, we must generate a *mapping* between the atom and the solar-system domains, recognising that the sun is the counterpart of the nucleus, and the planets match up with the electrons. Fourth is the optional *adaptation* phase where source domain information without a counterpart in the target, is transferred and adapted to the target domain. Thus, we learn that electrons `revolve-around` the nucleus, just like the planets `orbit` the sun. Finally, analogies support the *induction* of general rules, and the solar-system:atom analogy might suggest a generalised "central force" solution. This might indicate that smaller objects revolve around larger objects that they are attracted to.

We use a five-phase model of analogy adapted from Keane (1994), which recognises *representation, retrieval, mapping, validation* and *induction* (see Figure 1.2). Others like Kokinov (1994) identify different post-mapping phases like *transfer* and *learning*. Hall (1989) compares a number of models under the phases of *recognition, elaboration, evaluation*

and *consolidation*; Holyoak and Thagard (1989) recognise four phases of *retrieval, mapping, transfer* and *subsequent learning*. A number of authors identify three phase models; Eskeridge (1994) recognises *retrieval, mapping* and *transfer and use;* Eliasmith and Thagard (2001) identify *retrieval, mapping* and *application,* Falkenhainer, Forbus and Gentner (1988) identify phases of *access, mapping,* and *evaluation and use.* Interestingly, Hadamards' decomposition of creativity into the phases of *preparation, incubation, illumination* and *verification* is reminiscent of some of these phase models (Boden, 1992).

Figure 1.2 - *A Five-Phase Model of Analogy*

Computational modelling has helped bring broad agreement on the phase structure of analogy, and particularly on the mapping and retrieval phases. This research has helped identify a large number of constraints on the analogy process, ranging from working-memory limitations to neurological considerations (Keane et al, 1994; Hummel and Holyoak, 1997). Analogy has also played a well-documented role in scientific discovery (Koestler, 1964; Boden, 1992; Gentner *et al*, 1997). However, computational modelling of scientific analogies has largely focused on understanding a given analogy, rather than understanding how these analogies were discovered. In contrast, we focus on the problem of, for a given target problem, how can we find a novel analogy for it that supplies useful inferences to that problem, and how might we support this process in a computational model?

A computational model that can find novel analogies will encompass at least three phases of the 5-phase model in Figure 1.2. Firstly, it must *retrieve* appropriate source domains that can form a mapping with the target, and supply inferences to it. Secondly, it must include a model of the core *mapping* phase, to identify the correspondences between the two domains. Finally, it must *validate* the inferences that are mandated by the analogy. The central mapping phase has long been a focus for the computational modelling

community (Keane and Brayshaw, 1988; Falkenhainer *et al*, 1989), and many models of this phase exist. However, both retrieval and validation have received comparatively little attention from the computational modelling community, and these two phases will be the primary concern of this thesis.

Retrieving a novel source domain is crucial to finding a novel analogy. This may be partly attributed to the fact that all the similar sources have already been investigated, and are no longer considered novel. Therefore, creative retrieval must allow semantically distant sources to be identified, but should favour domains that might form a useful analogy with the given target. The mapping model then, must generate the mappings between these semantically dis-similar domains, and construct the mandated inferences. Finally, the validation phase must ensure that the conclusions drawn from the analogy are credible. Validation can even determine whether the entire analogy is accepted or rejected. We will use examples to highlight the role that validation plays in deriving the correct interpretation of many analogies. In particular, we will focus on the implicit dependency between the phases of retrieval and validation.

1.3 Finding Valid Analogies

The goal of this project is to develop a computational model capable of finding novel analogies for some given target problem. The model we wish to create should identify sources that provide a new interpretation of the target problem, as well as supplying novel inferences to that domain. As stated above, this will require interaction amongst the three central phases of analogy (Figure 1.3). The following examples highlight that even the relatively straight-forward task of interpreting a *given* analogy, can rely heavily on interactions between these phases.

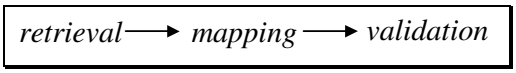

Figure 1. 3 - *Our 3 Phase model for Finding Novel Analogies*

1.3.1 Retrieval, Mapping and Validation in Within-Domain Analogies (Numeric)

To highlight the inter-play amongst these central phases of analogy, we start with a numeric source and target domain. When both source and target are from the same domain, we refer to these as *within-domain* analogies (or "near transfer" comparisons; Perkins and Salmon, 1992). Consider the inherent ambiguity of the following numeric analogy (Hofstadter and Mitchell, 1990):

`1:2 :: 3: [4]` or alternatively

`1:2 :: 3: [6]`

These proportional analogies are read as "one is to two, as three is to four" where the "4" is generated by the analogical comparison. Retrieving different information from memory supports alternate interpretations of the source. Each interpretation results in a different relation being added to the target. Identifying `one-plus` as the source relation creates one analogy, while accessing `half-of` creates a different comparison. The validation phase is responsible for ensuring that the identified source relation will never be applied to the wrong target values. So on a target domain that already contains the values 3 and 4, validation would ensure that the `half-of` relation is not applied to these values. Thus, the correct understanding of the analogy can depend on rejecting any interpretation that generates an invalid inference. This rejection may be followed by uncovering the correct source interpretation and generating the correct analogy.

The way in which the analogy is described, can compound the ambiguity in the source domain description. Let us consider the previous numeric analogy again, but now highlighting that there are two ways to present even this simple analogy. Restating it in a different format (A:C :: B:D rather than A:B :: C:D above) influences the highlighted relationship, and thereby the mandated inference.

`1:3 :: 2: [6]`

This highlights that the manner in which the analogy is described can influence the inferences that are drawn. Interspersing source and target information when describing the analogy, can highlight different information and cause different inferences. Thus a poorly presented analogy may

introduce ambiguity and lead to unwelcome inferences, which *should* be rejected before eventually arriving at the intended interpretation.

Natural analogies may contain significantly more ambiguity in the description of the source, thereby placing a greater onus on validation to arrive at the correct interpretation. The source description is embedded within its domain, from which alternative or additional information may easily be brought forward. Validation must be eternally vigilant against any injurious inferences that might result. As an example of a badly described natural analogy, consider the following:

`cow:chicken :: calf:[egg]`

No identifiable source relation (such as "frightens") can reasonably be applied to the target (an `egg` isn't really something that is going to be frightened). However, the correctly described analogy using the same objects, but presented in a different way, is as follows.

`cow:calf :: chicken:[egg]`

This highlights a source relation that *is* applicable to the target domain. This order of presentation may be related to the order effects identified by Keane (1997).

1.3.2 Retrieval, Mapping and Validation in Between-Domain Analogies (Numeric & Alphabetic)

We now examine validation in analogies that draw on two different domains of knowledge (e.g., numeric and alphabetic), the so-called *between-domains* analogies (or "far transfer" comparisons; Perkins and Salmon, 1992). Consider the following:

`* 1:2 :: a:[b]`

If either of the numeric relations identified above (`one-plus` and `half-of`) were retrieved and applied to the target, a nonsensical inference would result. One needs to identify the source relation `1 predecessor-of 2` to generate the correct inference. Here, rejecting the invalid inferences appears to operate as a restriction on the inter-domain mapping, ultimately rejecting the more obvious source predicate `1 one-plus 2` by the resulting inference. We do not even require a great deal of mathematical

knowledge to reject either the `a one-plus b` or `a half-of b` inferences, as these mathematical relations require numeric arguments.

When they are being interpreted by a computational model, these between-domains analogies are far more likely to generate invalid and nonsensical inferences than within-domain analogies. Injecting source material from one domain into a semantically dissimilar target, may generate incongruous combination of source and target material. Detecting, rejecting and adapting these inferences is the responsibility of the validation phase. The unusual combinations of source and target information that between-domain analogies can create, may even make their rogue inferences easier to detect than the more plausible within-domains inferences. In contrast, within-domains analogies are much less likely to generate such incongruous inferences, and place a different requirement on the validation stage.

1.3.3 Historical Examples of Finding Valid Analogies:

Kekulé's Discovery

Scientific analogies also rely on interactions between the retrieval and validation phases. Creative scientific analogies generally arise from between-domains comparisons (Boden, 1992), and some of these creative analogies are well documented - although the origin of the source domain is difficult to ascertain. Hoffman (1995) highlights that many scientific breakthroughs can be neatly summarised by the creative analogies upon which they are based. The following creative analogy highlights the interaction between the retrieval, mapping and validation processes.

Let us consider the analogies used by Kekulé that explained the behaviour of most organic compounds. Kekulés initial 1855 analogy was between a Carbon atom and a link in a chain (Boden, 1992). A carbon-carbon bond often forms large sequences of Carbon atoms, as shown in Figure 1.4. This "carbon chain" analogy explained both the structure and reactive properties of most organic compounds, and effectively marked the birth of organic chemistry.

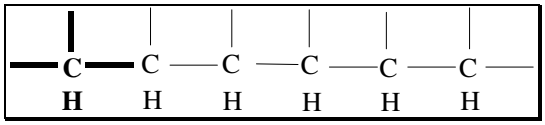

Figure 1. 4 - *Kekulé's Carbon Chain*

Significantly, this creative analogy relied on accepting the *novel* inference that a Carbon atom can form a bond with another Carbon atom. This novel inference was crucial to accepting the carbon-chain analogy, being described as two carbon atoms "dancing in the street". So we can theorize that validation would have played a crucial role in accepting this analogy, because of its novel inference; and perhaps in explaining why previous interpretations were deficient. However, Kekulés next analogy provides clearer evidence for the role of validation.

The behaviour of the C_6H_6 molecule (in particular) contradicted the behaviour expected under the carbon chain theory, which suggests a highly reactive material with many unused bonds (Figure 1.4). Yet C_6H_6 was observed to be highly stable. Ten years passed before Kekulé's next analogical insight - which arose while thinking of the carbon chain as a snake biting its own tail (Koestler, 1964; O'Donoghue, 1997, 1999). This created the famous "carbon ring" structure, whereby the ends of a C_6H_6 molecule meet each other. This creative analogical comparison effectively marks the birth of aromatic chemistry.

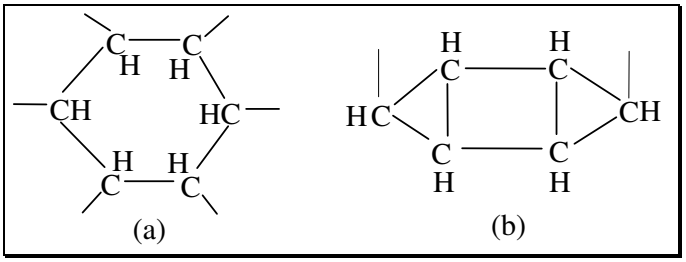

Figure 1. 5 - *Rejected versions of Kekule's Carbon Ring*

So, why was there a ten-year wait for the carbon ring analogy? One possibility is that in deriving the final format of the carbon ring analogy, Kekulé first had to reject a number of other versions of the carbon ring –

involving the processes of analogical validation and adaptation. The initial structure suggested by this analogy is depicted in Figure 1.5 (a) - reducing the number of unattached bonds from 8 to 6! Adding the earlier novel inference that Carbon can bond to two other Carbon atoms, brings us to the structure in Figure 1.5 (b) - with just two unattached Carbon bonds. The final novel inference relied on allowing double bonds between Carbon atoms, Figure 1.6.

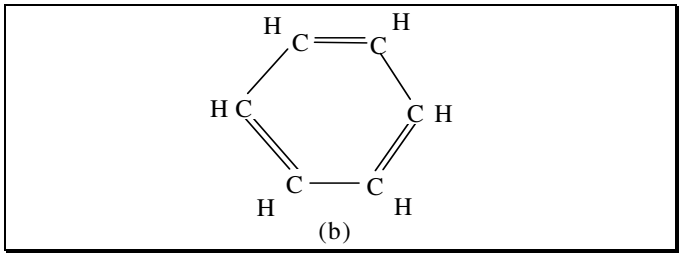

Figure 1. 6 - *Kekule's Carbon Ring*

Thus, while analogically driven, many inferences had to be rejected by some process of validation. This resulted in a sequence of interpretations of the carbon-ring analogy, before the final interpretation explained all known phenomena, including those explained by the previous theory. Without a validation process, only the initial interpretation would have been investigated, before being swiftly rejected. So, validation may well have played a crucial role in sustaining and developing the initial idea, before the correct interpretation of the comparison was reached.

Duncker Example

We can also observe the inter-dependency between the retrieval and validation phases in Duncker's (1945) analogy between a malignant tumour and a heavily defended fortress. The source domain involves a fortress that is conquered by sending troops down different roads to converge on the fortress, thereby conquering it. This serves as the source domain for an inoperable tumour, which is treated by using multiple x-rays - all of which converge on the tumour.

Now consider accessing some additional source domain information, such as the General instructing his soldiers to inspect their rifles. This generates the inference that the counterpart of the soldiers, inspects their

rifles – namely x-rays inspect their rifles. Clearly, this inference is not directly applicable to the target domain and should be rejected by the validation phase (though adaptation could potentially modify the rejected inference to better fit the target domain). An overly zealous retrieval process may place a heavy burden on validation, while a lethargic retrieval process will not make full use of the available analogical comparisons.

1.3.4 Retrieval and Inference

From the previous examples we can identify two distinct ways in which information enters an analogy. We use the term *access* to indicate retrieval from a denoted source domain, while we use the term *retrieval* to denote spontaneous retrieval of a novel (candidate) source domain. When interpreting a given analogy, validation must ensure that the retrieval phase does not *access* any information that is not applicable to the target domain. Similarly, when retrieving a novel source domain, validation must ensure that the entire analogy is valid. So while the objective of access differs from retrieval, many of the implications for validation are the same. When searching for a novel source domain, one may expect a greater number of validation rejections than when accessing a recommended source.

From the previous examples, it would appear that validation is carried out largely within the target domain, and that this requires a degree of familiarity with the subject matter of that target. It seems natural to suggest that a familiar target domain can support more accurate validation than an unfamiliar target. For example, in the "marathon is like an examination" analogy, "cramming" all night before an examination might suggest that training all night before a marathon is a useful way to prepare for such a race. Only "common sense" understanding of the marathon domain will reject this inference. However, we may also be able to perform validation in relatively unfamiliar target domains, based on a detailed understanding of the inferences that originate in the source - and its interaction with the target-based information. Let us briefly reconsider the earlier between-domains analogy `1:2 :: a:b`. Rejecting the `1 one-plus 2` predicate did not require a large amount of knowledge on either numeric or alphabetic

concepts. That is because the inference this analogy suggests, would create an incongruous combination of numeric and alphabetic concepts. The most obvious way of rejecting the `one-plus` interpretation may be by validating (and rejecting) the inference that this would generate.

1.3.6 Analogy and Creativity

Creative analogies are of particular relevance to this thesis, because finding a new analogy is typically seen as a creative process. Creative analogies generally originate in between-domain comparisons (Boden, 1992), providing a new perspective on the problem domain. Creative analogies are sometimes well documented, with both the previous and resulting analogical interpretations being recorded.

Analogy is central to many recorded creative experiences (Boden, 1992), particularly scientific ones. There is often an extensive search for a suitable source domain, leading to the "serendipity" view of creative retrieval. However, creative scientists generally spend many years exploring different analogies before this "serendipity" takes place (Curie, 1923). Many of the analogies that are explored before serendipity occurs, generate invalid inferences and must be rejected by the validation process. We will examine some creative analogies in this thesis, as they require extensive use of both the retrieval and validation processes.

We adopt Boden's (1992) terminology when describing analogies, and refer to historically creative concepts as *h-creative*. These have never been used before by anyone, whereas *p-creative* (psychologically creative) concepts are new to the reasoning agent. Of course, an *h-creative* analogy will also be *p-creative*. We use these two terms to describe both the analogical comparisons, and to the inferences that they create. Ritchie (2001) also highlights *novelty* as one of the essential qualities of creativity.

1.4 Conclusion

Reasoning by analogy is a very complex and flexible mechanism that is closely linked with many other cognitive processes. We present an interpretation of analogy as a flexible reasoning process, whose primary function is to cause learning. Interpreting a given analogy (even a simple

alpha-numeric one), can require a surprising amount of interaction among the central phases of analogy. Retrieval must ensure that all relevant information is retrieved from the source domain, and brought into the analogy to generate the desired inferences. But validation must be eternally vigilant that retrieval does not identify inappropriate information, because this could lead to invalid inferences. However, our focus is not on interpreting given analogies.

The objective of this thesis is to create an analogy-based system capable of finding novel analogies for some given target problem. In constructing this system, we will present two complimentary models. The first is a new analogy retrieval algorithm that identifies domains with the potential to generate inferences for some given target. The other is the validation mechanism, which will examine the inferences sanctioned by each of these comparisons. A mapping model will connect the retrieval and validation models, forming a contiguous model encompassing three phases of analogy.

Retrieval will therefore act as the driving mechanism to test the validation process, which either accepts or rejects the resulting inferences. Validation then, will operate as a restriction on the retrieval process, eliminating any identified sources that generate invalid inferences. We can think of the combined system as a simple creativity engine (Boden, 1998), searching for sources that generate reasonable inferences about some given target.

1.4.1 Goal of this Thesis

In Chapter 2 we review previous models of analogical retrieval, mapping and validation. We assess any implications for our model, and examine the structures that are used to support these models.

In Chapter 3 we describe the model of memory that underpins much of the model we will propose. The memory model plays an important role in both retrieval and validation.

Chapter 4 introduces the new models of the retrieval and validation phases. It also includes a brief description of the mapping model that was used.

Chapter 5 is the first of two results chapters that assess these models. This chapter assesses the model of analogical retrieval, by discussing and assessing some retrieval tests that were performed.

Chapter 6 assesses the model of analogical validation. It describes similar experiments to those in Chapter 5, discussing and assessing the performance of the validation model.

Finally, Chapter 7 examines the overall success of the project.

Chapter 2

Background

"Experience is what you get when you don't get what you want."

-- Anon.

"Life is the art of drawing sufficient conclusions from insufficient premises",

- -Samuel Butler, Notebooks, Life ix. 1912

2.1 Introduction

Our model for generating new analogies must identify novel source domains that provide new inferences for some target problem. To achieve this goal, the model will encompass the three phases of *retrieval, mapping* and *validation*. In this chapter we will examine a selection of existing analogy models that address these three phases. This review will describe the operation of each model, and will assess its potential for contributing to the creative analogy model.

Gentner's Structure Mapping Theory (1983) inspired early work into modelling the mapping phase of analogy (Keane and Brayshaw, 1988; Falkenhainer, Forbus, and Gentner, 1989; Holyoak and Thagard, 1989).

Much of this and subsequent work on mapping has served to highlight specific influences on the analogy process (Kokinov, 1994; Eskridge, 1994; Veale, 1995). The inter-domain mapping is also the key to generating inferences, and is typically modelled as a form of pattern completion (Holyoak, Novick and Melz, 1994). Models were later developed to include the retrieval phase (Gentner and Forbus, 1991; Thagard *et al*, 1990; Plate, 1998; Eliasmith and Thagard, 2001). Work also progressed on the post-mapping activities including inference verification (Falkenhainer, 1990) and induction (Hummel and Holyoak, 1996). Thus, research gradually built up a more complete picture of the entire analogy process. Although none of this work focused specifically on creative analogising, these models may provide valuable insight for our present work.

Researchers have adopted a surprising variety of modelling techniques, some of which may be appropriate to our requirements. Initial models operated deterministically (Keane and Brayshaw, 1988; Falkenhainer, Forbus and Gentner, 1989), but stochastic (Hofstadter and Mitchell, 1990) and parallel constraint satisfaction models (Hummel and Holyoak, 1989) have also been developed. Our creative model will require a memory of domains from which to find an appropriate source, and some analogy models have been developed to deal with specific memory structures. These include localist-connectionist memories (Veale, 1995), and a more neurologically inspired framework (Hummel and Holyoak, 1996).

The objective behind reviewing these models is to identify useful lessons for our model of creative analogising. Of course, we will also identify any potential pit-falls as they arise. Our review will pay particular attention to the ability of retrieval models to identify semantically distant sources. When examining mapping models, we will pay particular attention to their ability to form mappings with semantically dissimilar sources. Finally, we will assess the suitability of validation models to detecting the diverse range of invalid inferences that we might expect from a creativity engine.

2.1.1 Structure of this Chapter

This chapter examines a large number of models of the analogy process, all of which include a model of the central mapping phase. These models are grouped so that multi-phase models are examined together as a collection. We begin with the multi-phase models, addressing retrieval, mapping and validation. We then progress to some of the smaller uni-phase models. Because just a few models addressed verification and induction, these will be given special attention when they are encountered.

At the end of the chapter we summarise the various approaches taken to modelling analogy. The chapter finishes with a description of the requirements that are placed on our model for finding novel analogies.

2.2 SME And Its Associates (MAC/FAC and Phineas)

The first suite of models that we examine was developed around the best-known model of analogical mapping, the Structure Mapping Engine - SME (Falkenhainer, Forbus and Gentner, 1989). The SME mapping model has been augmented by models of retrieval and of verification. We begin by examining the SME model before examining the MAC/FAC (Forbus and Gentner 1991; Forbus, Gentner and Law, 1995) model. MAC/FAC is a two-part model, the first part is a retrieval model called MAC (Many Are Called) and the second part FAC (Few Are Chosen) is essentially a version of SME. Finally, we examine the Phineas (Falkenhainer, 1987; 1988-b) model encompassing retrieval, mapping and verification.

2.2.1 SME - The Structure Mapping Engine

SME (Falkenhainer, 1988-a; Falkenhainer, Forbus and Gentner, 1986, 1989) is an implementation of Gentner's (1983) *Structure Mapping Theory* (SMT) of analogy. SMT specifies *what* needs to be computed in processing analogy, while SME specifies *how* this computation is carried out. SMT highlights three constraints on the types of inter-domain mappings that may be formed. First, *relational consistency* dictates that items in one domain can be mapped to 1 and only 1 item in the other domain. Secondly, *parallel connectivity* says that if two items are related in the source, then their mapping counterparts

must also be related in the target. Finally, *systematicity* says that analogies favour large systems of relations, connected by high-order relations.

SME provided the original implementation of SMT, and has since undergone successive alterations. Algorithmically we can see the original algorithm as performing an optimal search, while successive improvements performed a greedy search and incremental solution generation. However, we shall describe the essence of SME before describing these algorithmic variants.

To illustrate our description, we will make use of Rutherford's famous *atom:solar-system* analogy (Gentner, 1983). In this analogy the *solar-system* is the well understood domain and is referred to as the *source*. The *atom* domain is the one we wish to learn more about, and is referred to as the *target* domain. In SME the fundamental collection of domain information is the Description Group (*Dgroup*), containing all relations and objects for that domain. Dgroups contain entities (representing objects and constants) that are typically connected to form predicates. First-order predicates describe relations between objects, while high-order predicates represent causal relations between first-order predicates (or other high-order predicates). The two Dgroups for the *atom:solar-system* analogy are depicted in Figure 2.1, one Dgroup represents the solar-system source and the other represents the atom target.

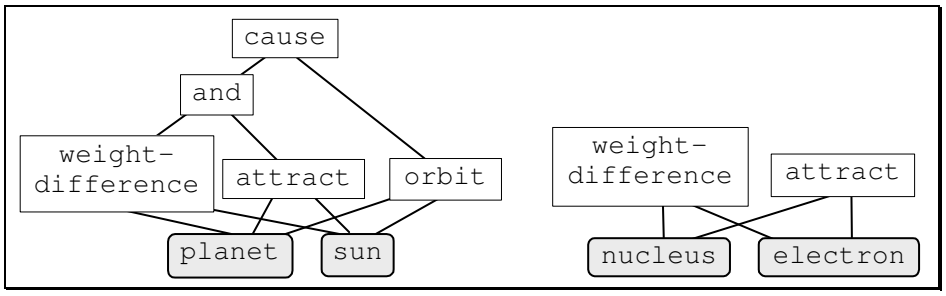

Figure 2. 1 – *Domains from the atom: solar-system analogy*

SME is a four-part search algorithm, gradually composing larger structures until the best inter-domain mapping is constructed. The phases of SME are:

I. *Local Match Construction*: SME first generates all allowed matching pairs, which may or may not form part of the final interpretation.

II. *Global Match (gmap) Construction:* Combines the matching pairs into the largest collection of paired entities, while conforming to the constraints of SMT.

III. *Candidate Inference Construction*: Derive the inferences suggested by each mapping.

IV. *Match Evaluation*: Evaluate the goodness of each mapping together with the suggested inferences.

Now we shall examine each of these four stages in more detail.

Stage I, *Local Match Construction,* generates the elementary-level pairs of source and target items, from which the final mapping will be composed. This process constructs a *match hypothesis* (MH) for each item in each domain of the form (<source-item>, <target-item>), but only those pairs that are allowed by SMT are generated. So, each source domain item is matched against all valid counterparts in the target domain. The constraints guiding the construction of these match-hypothesss are: first only *identical* predicates can match; second, items match if their predicates are *functions*; and third, items match if they are both *objects* or *constants* and the comparison is supported by some other identity. This stage also calculates evidence scores for each MH, by examining the structural and syntactic properties of that match. The evidence score for an MH increases when some high-order structure supports that match. Table 2.1 below lists all the match-hypotheses constructed for the *solar-system:atom* example (Figure 2.1), together with their evidence scores.

In Stage II *Global Match (gmap) Construction*, individual match-hypotheses are combined to form different interpretations of the inter-domain mapping. Gmap construction begins by constructing partial matches (*pmap*). A *pmap* is are collections of MHs that are built on top of an individual MH, plus all other MHs that are structurally implied by that MH (Figure 2.2). So, for a *pmap* that is built on top of an HM representing the mapping between two relations, the *pmap* will include this MH, the MHs representing the matched arguments, plus MHs representing any high-order relations that use

the original MH as an argument. An individual MH might participate in several different *pmaps*, with each *pmap* typically containing a small portion of the domain information.

MH	Evidence Score
(weight-difference, weight-difference)	0.7900
(attracts, attracts)	0.7900
(planet, electron)	0.8646
(sun, nucleus)	0. 8646

Table 2. 1 - *Match Hypotheses for the solar-system:atom analogy*

After these pmaps are constructed, *gmaps* are constructed by merging maximally consistent collections of pmaps. Each gmap represents the largest compatible collection of match hypotheses that can be constructed, without violating the SMT constraints of *relational consistency* and *parallel connectivity*. Each gmap represents a different interpretation of the analogy, and Stage II often creates several competing gmaps. Gmap construction involves an exhaustive search of the possible *gmaps* to ensure the largest *gmap* is found. The "standard" interpretation of the *solar-system:atom* analogy is listed in Table 2.2.

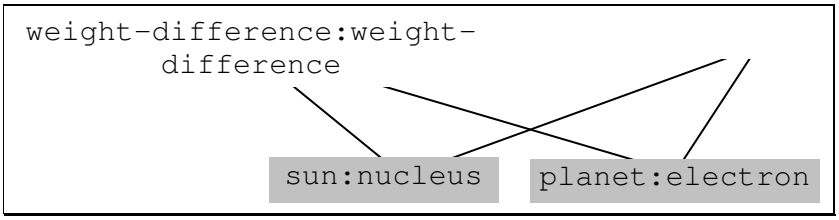

Figure 2. 2 - *Match Hypotheses Combine to Form Pmaps*

Stage III is *Candidate Inference Construction,* candidate inferences are made to represent the inferences implied by the analogy. Technically, these candidate inferences are source domain items for which there is no mapping

in the target domain. Information to be "carried over" to the target is identified by a process of pattern completion operating on the inter-mapping. The candidate inferences in our example are based on the `and, orbit` and `cause` relations from the source domain. This pattern completion process generates the candidate inferences: `orbit(electron, nucleus)` and `and(weight-difference orbit)`. (Note: SME did not generate the causal inference, as it is not structurally grounded - see Table 2.3).

Gmap	**Weight**
`(weight-difference, weight-difference)`	
`(sun, nucleus)(planet, electron)`	
`(attracts, attracts)`	3.3092

Table 2. 2 - *SME Generates Only One Gmap for the Solar-system:Atom Analogy*

A candidate inference must satisfy some purely structural constraints, or it is rejected. First, the sub-expressions of the inference must intersect the underlying gmap - this constraint is known as "structural grounding". So in our example, the `orbit(electron, nucleus)` inference intersects with the underlying gmap because both arguments participate in the gmap (Table 2.3). Second, if the predicate is non-commutative, then its arguments cannot be permuted versions of the arguments to another expression in the target. If the unmapped source item is a simple constant (*eg* `zero`) then it is transferred directly into the target domain. Otherwise SME introduces a new hypothetical entity to the target as a skolem[1] function of the original source item `skolem(source-item)`.

[1] A Skolem object represents an object whose properties are known but whose (exact) identity is not known. While analogical transfer might suggest the presence of new objects and relations in the target, the exact identity of these objects and relations may not be derived directly from the analogical comparison. Skolem

Candidate Inferences:
(and weight-difference attracts)

Table 2. 3 - *SME Identifies One Inference for the Solar-system:Atom Analogy*

In Stage IV, *Match Evaluation,* the "best" inter-domain mapping is selected. SME computes a structural evaluation score (*ses*) for each gmap. Belief values are propagated between individual match-hypotheses, under the control of a belief maintenance system - but in many cases these results are similar to those you would expect from a simple "weighted sum" of individual match-hypotheses. In our working example, this would select the gmap of Table 2.3 plus the inference in Table 2.4.

Gmap-1: (weight-difference, weight-difference)
(sun, nucleus)(planet, electron)
(attracts, attracts)
Candidate Inferences:
(and weight-difference attracts)

Table 2. 4 - *The Final Output from SME*

Algorithmic Variants on SME

The original and optimal version of SME (Falkenhainer, Forbus and Gentner, 1989), referred to here as O/SME, identified the largest possible inter-domain mapping for source and target pair. O/SME performed an exhaustive search through the space of all possible gmaps, to identify the largest Gmap. The guarantee this algorithm gives of finding all the maximally sized mappings between domains, causes serious scalability problems. O/SME effectively finds the "Largest Common Subgraph" between two domains (Veale, Smyth, O'Donoghue and Keane, 1996), a problem that is recognised to be NP-Complete (Garey and Johnson, 1979). The cost of optimality effectively

---

entities are very closely related to existentially quantified variables (Clocksin and

limits O/SME to solving only smaller mapping problems. However, its authors claim that this was not a problem in practice - particularly when significant amounts of domain structure were present (but see Veale and Keane, 1997).

Variant I - Greedy SME

The real "bottleneck" of the O/SME algorithm centres on the gmap construction step (Stage II). To overcome its complexity problems a heuristic *greedy merge* version of SME was developed, referred to here as G/SME (Forbus and Oblinger, 1990). The key novelty in G/SME is the notion of a *kernel* predicate that is a root concept within a domain description representing a predicate that is not an argument to any other predicate. So in the solar system-domain the `cause` relation is a kernel predicate. The atom domain has two kernels as neither `weight-difference` nor `attracts` are arguments to another predicate. Kernel predicates can often be thought of as the controlling causal relations within a domain. A kernel mapping then, is a mapping between kernel predicates from each domain, and includes all subsequent mapping below this top-level match-hypothesis. By definition, each kernel mapping generates a large consistent set of match-hypotheses. G/SME starts with the largest kernel mapping and then folds in other kernels as long as they do not violate structural consistency. By favouring the combination of kernel mappings with large structural evaluation scores, G/SME significantly reduces the search space of *gmaps*. However, G/SME does not guarantee that maximal sized mappings will be found, and so useful alternative interpretations might be overlooked.

Variant II - Incremental SME

SME's most recent incarnation, referred to here as I/SME (Forbus, Ferguson and Gentner, 1994) uses an *incremental* mapping algorithm. I/SME borrows very heavily from IAM (Keane and Brayshaw, 1988), which models the way in which analogies can be compiled or added to when new information becomes available. As pointed out by its authors, when all domain information is made available simultaneously, the output of I/SME is the

---

Mellish, 1984).

same as G/SME. I/SME is useful when new domain information becomes available, to extend the current inter-domain mapping. For example, if we discovered that the attractive force between the sun and planet is proportional to their weight difference, we would like to add this extra information into the current mapping. This might map with available information in the atom domain, or may result in generating some new inferences. Given new source and target items Si and Ti respectively, and a previous global mapping Mi (if any), then I/SME operates as follows.

1) I/SME extends the set of local match hypotheses MHi by testing Si against new and old target items, and similarly by testing Ti against new and old source items. <Si, Ti> is only added if its addition does not break the 1:1 mapping restriction.

2) Update the set of Kernels starting with the new root match-hypothesis, and search downward for the first structurally consistent match-hypothesis. This match-hypothesis and its descendants form a new kernel, where new match-hypotheses can generate new kernels or attach as part of old kernel structures. We now have a new set of kernel structures to add to the overall mapping.

3) Perform a structural evaluation on the new MH's and on the set of new kernel structures. I/SME also allows pragmatic filtering of these new kernels at this stage.

4) Extend the set of global mappings Mi by merging members of the set of new kernels with Mi's. If no Mi's exist then the greedy algorithm generates the initial set of Mi's from the set of kernel mappings.

Also, SME provides a *remap* operation that may be invoked by some external process, should the derived mapping be unsuitable for some problem. Performing a *remap* allows backtracking over the existing mapping at the level of kernels, allowing an alternative mapping to be generated that might meet the needs of the external process. This increases the flexibility of the analogy system by recombining match-hypotheses to generate a new set of correspondences between source and target. So, all versions of SME model the core mapping phase of analogy.

2.2.2 MAC/FAC

SME has a counterpart model that addresses the retrieval phase of analogy. Many Are Called but Few Are Chosen (MAC/FAC, Gentner and Forbus, 1991; Forbus, Gentner, and Law, 1994; Law Forbus and Gentner, 1994) is a model of analogical retrieval (and mapping) that incorporates the SME model of mapping. MAC/FAC is a two-stage algorithm (see Figure 2.3), where the initial MAC stage scans memory using a description of the target domain to perform a similarity-based selection of candidate sources. MAC first finds candidate sources whose descriptions resemble the given problem domain. MAC/FAC then uses SME for its second FAC stage, to generate a detailed structure mapping between the target domain and the selected sources. The best of these candidate sources is chosen by MAC/FAC as an analog for that one domain.

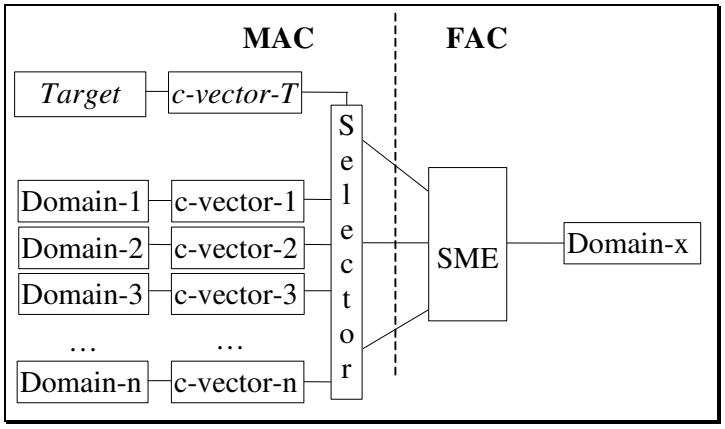

Figure 2. 3- *MAC and FAC Stages of Analogical Retrieval*

The MAC stage describes each domain with a content vector (*c-vector*), which is an n-tuple of numbers representing the quantity of each element in that domain. So, both the *solar-system* and *atom* domains contain one instance of both `weight-difference` and `attracts`, while the solar-system additionally contains one instance each of the `orbits`, `and` and `cause` relations see Table 2.5. Domains that contain multiple instances of a predicate will result in *c-vectors* containing larger integer values. Using this

c-vector representation means that MAC performs semantic based retrieval using a token identicality mechanism.

Content Vector:	(weight-difference, attract, orbit, and, cause)
Solar-system:	(1, 1, 1, 1, 1)
Atom:	(1, 1, 0, 0, 0)
Dot Product:	2

Table 2. 5 - *Content Vectors for Solar-system and Atom*

Candidate source selection then involves computing the dot product between the target's c-vector, and the c-vector of every source in memory. So the similarity score between these two domains is the dot-product of two corresponding c-vectors. The solar-system domain will be returned if it is within 10% of the best identified source across all sources. 10% is effectively an arbitrary threshold used by MAC to select the most promising sources for further investigation. Computing the dot product is a relatively inexpensive operation to perform for each source, and the MAC stage scales linearly with memory size. So candidate source selection is effectively based on an intersection search using the semantic-features of the target and the source. Once the FAC stage has identified a set of candidate sources, SME performs a detailed comparison between the target and each candidate source in turn. After evaluating all candidates, SME selects the best sources with the greatest degree of structural similarity (and the highest structural evaluation score) from the candidate sources.

We illustrate the operation of MAC/FAC again using the solar-system and atom example. The target contains exactly one instance of each of the following semantic tokens: `weight-difference`, `and` and `attracts`, along with two objects, a `nucleus` and an `electron`. So the MAC stage will search for sources that also contain this same information. The desired solar-system source contains the same predicates but has two different objects. The source also contains two additional relations, *cause* and *orbit*. The dot-product between these content vectors allows the solar-system

domain to progress to the SME stage, for more detailed structure mapping. Content vectors summarise the semantic content of domain information, and thus the first cut domain selection is based on its semantic content and disregards its structure. So, MAC retrieves sources that share identical tokens with the target, before invoking SME (FAC) to perform a more detailed structure mapping.

2.2.3 Phineas

Having looked at the mapping and retrieval models, we now examine the associated model of verification-based learning called Phineas (Falkenhainer, 1987, 1988-a, 1990). Phineas was developed specifically for the domain of physical analogies, such as "heat-flow is like water-flow". Specifically, Phineas addresses the problem of finding qualitative explanations of time-varying physical behaviours. Its objective is to find new interpretations of a given target domain that might form the basis of further understanding and hypothesis formation. Because Phineas is one of the rare models to address the post-mapping validation (or "verification") phase, we will examine it in detail.

Representations of Knowledge in Phineas

The physical domains that Phineas deals with are described with a special notation called *qualitative process theory* (QP) (Forbus, 1984). QP theory represents situations as collections of objects, relationships, and a set of process schemata that account for changes in that scenario (domain). Phineas uses three types of information during its reasoning. First, the *initial domain theory* contains information about some physical situation, and is described using QP theory. Second, a library of *prior experiences* contains structural and behavioural descriptions that summarise information on the attributes that participate in a scenario. Observations that occur at the levels of the scenario can represent: state information `open(jar)`, behaviours `decreasing (amount(alcohol))`, and behavioural abstractions `asymptotic`, `continuous`, and `invariant-movement`. Finally, Phineas records *observations* that are targeted for explanation. Phineas seeks

to produce explanations formed from a set of process descriptions, entity descriptions and atomic facts.

Access in Phineas

Phineas uses a four-stage algorithm encompassing *access, mapping and transfer, qualitative simulation,* and *revision* (see Figure 2.4). *Access* is a two-part process that retrieves candidate sources from memory when it is presented with a target problem. Access is based on identifying common *behavioural abstractions* between the source and target (like `asymptotic`, `continuous`). These behavioural abstractions include role and other domain-specific information to retrieve candidate sources from memory. The second part of access uses SME to generate a mapping between the behavioural abstractions of the two domains. At this stage SME operates only at the abstraction level, identifying the roles that each element plays within the abstraction. By performing matching at this level, SME notices that the role played by the `nucleus` is similar to that of the `sun` it its abstraction layer, and that the `electron` plays a similar role to the `planets`. The output of the first part of the Phineas algorithm is a mapping at the abstraction layer level, between the two domains.

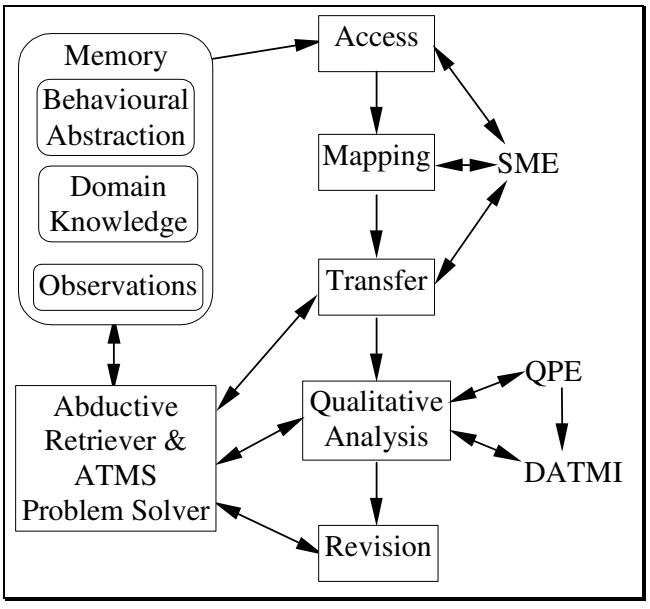

Figure 2. 4 – *Overview of the Phineas Model*

Mapping and Transfer in Phineas

The second stage of Phineas is the *mapping and transfer* process. In this stage, SME is again invoked to complete the mapping between the two domains. This is required because the first phase identifies the correspondence only at the abstraction level, and this is effectively a partial inter-domain mapping. SME now works in Contextual Structure Mapping (SME_{CSM}) mode, which allows the inclusion of role information in the mapping. SME_{CSM} ensures that corresponding roles and abstractions are mapped correctly between the two domains.

SME_{CSM} differs from standard SME in that it operates, not just on two domains, but also on the mapping between the abstraction layers that was created by the *access* stage. SME_{CSM} elaborates a given mapping between the abstract layers, by identifying the objects that occupy the various roles involved. SME_{CSM} tackles mapping from the perspective of roles, wherein objects and predicates are viewed as role-fillers within the domain abstraction. Because the number of objects filling a particular role may differ in the two domains, SME_{CSM} is capable of identify many-to-1 mappings. So, a single object in one domain may correspond to several objects in the other domain if the abstraction layers have been suitably described. Each analogy problem then, is seen as attempting to find corresponding role-fillers between the two domains. The final task of the mapping stage is inference generation (transfer), which is performed using SME's pattern completion model. Because of the information provided in the scenario descriptions, the inferences include information on behavioural abstractions, role descriptions, predicates and objects.

The Phineas algorithm is best described using Falkenhainer's (1990) thermostat analogy, which we use to illustrate Phineas' first two stages. Falkenhainer describes two alternative mechanical thermostats that are *mostly* analogous. The first thermostat (Figure 2.5 a) uses a coil that expands and contracts with heat, and this coil tilts to cause the mercury-based switch to turn on and off. When the furnace is on, the strip will expand and cause the mercury-based switch to turn off the valve – thereby regulating the

temperature. Adjusting the lever alters the minimum temperature at which the circuit is switched on. The second thermostat uses a bi-metallic rod to directly regulate the flow of gas to the furnace (Figure 2.5 b), by causing the bi-metallic strip to expand and cut off the supply of gas to the furnace. Adjusting the lever alters the minimum temperature at which the furnace will operate.

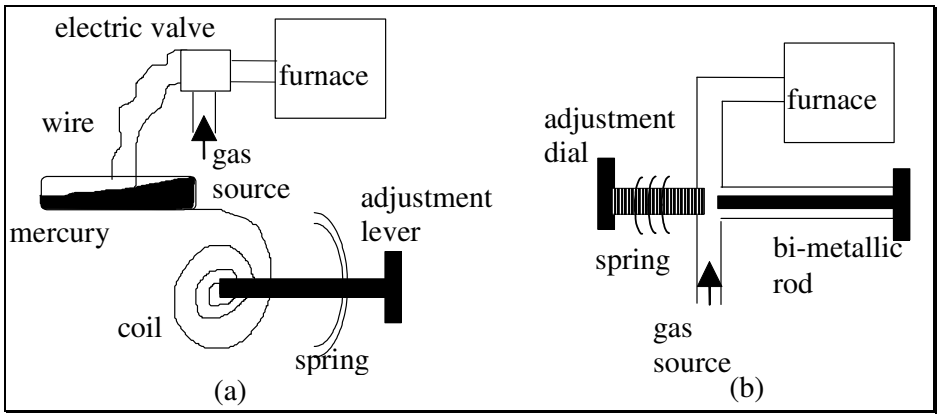

Figure 2. 5 - *Two Analogous Thermostats*

While there are some dissimilarities between them, Phineas identifies their many similarities. If we examine Figure 2.5 we can see that the manual adjustment levers play the same role in each domain. The gas and furnace play the same roles in each domain. The coil corresponds to the bi-metallic rod in domain (b). The mercury, wire and valve fulfil the same role in domain (a), as the rod and the adjustment lever in domain (b).

Qualitative Simulation in Phineas

The third part of the Phineas model is called *qualitative simulation*, and verifies the candidate inferences against the overall model of the target domain. The augmented target is passed to Forbus' *qualitative process engine* (QPE) for use in a form of gedanken experiment, creating an "envisionment" of the target domain under the new interpretation. If this experiment is complete and consistent with regard to known observations, then it is considered successful. Identifying points of discrepancy is performed by

DATMI (Dynamic Across-Time Measurement Interpretation), which is a program that relates the predictions generated by the qualitative theory to known observations of physical behaviour. Phineas uses the following operations to verify the candidate inferences against the overall domain theory:

- *Ask x.* This is a deductive process that tries to determine if the candidate inference (x) is deducible from target facts. If so, the candidate inference is verified for inclusion in the target domain.
- *Abductive-ask x.* This is akin to "ask x" but allows abductive inferences to be made in support of the candidate inference.
- *Retrieve-functional-analogue x.* This uses the role played by expressions, as a basis for identifying analogous expressions that can also fill the same role within the target domain.
- *Create-entity x.* When a Skolem object cannot be identified, a new object is created and is assumed to fill the corresponding role. This creates the entity and examines its proposed consistency, and if found to be consistent it replaces all instances of variable x. Inconsistency is deemed to be grounds for rejecting the analogy.

Revision in Phineas

The final phase of the Phineas model is *revision*. Revision is only required when the DATMI fails to verify the new interpretation against previously known information. Any conclusion that is rejected is sent to revision, which attempts to adapt information that was not successfully validated. Inadequate interpretations are adapted about their "points of inaccuracy", however this component has not been completely implemented.

Phineas is implemented as a pseudo-parallel model, with multiple explanations being maintained at various stages of development. A task agenda determines the relative priority of the processes, each task type operating upon an hypothesis, undergoing access or mapping etc. SME's similarity metric also helps determine the priority of various processes. When an hypothesis is accepted, a second preference criterion comes into play to discriminate between the accepted hypotheses. This uses the criteria of simplicity, plausibility, and specificity to select between the accepted

hypotheses. Determining simplicity, plausibility and specificity are themselves "open research problems", so this discrimination module has not been fully implemented.

2.2.4 Evaluation of SME and Associates

SME was the first successful implementation of the SMT theory, which proved the usefulness of SMT in interpreting analogies. SME has become the benchmark against which other mapping models are compared (Keane, Ledgeway and Duff, 1994). SME's pattern completion model for generating inferences has become the heart of the Copy With Substitution and Generation algorithm (Holyoak, Novick and Melz, 1994). The complexity problems associated with the original implementation were soon overcome by later versions of SME.

SME forms the core of a suite of models, encompassing the three contiguous phases of retrieval, mapping and verification. The MAC/FAC retrieval model complements SME by retrieving sources described by the same relations as the target, thereby identifying domains that may successfully be mapped by SME. Phineas was the first model to consider the post-mapping verification phase. Phineas illustrated that a "deep" model of verification requires a wealth of domain specific knowledge, comparing inferences to previously known target facts.

However MAC/FAC is not well suited to the task of identifying novel (*p-creative* or *h-creative)* analogies (Boden 1992). The strong role that predicate identicality plays in both models, severely constraints the types of candidate source and can be retrieved and mapped. For example, these models will not generate analogies with source domains that are described using synonymous relations to those in the target probe (So, the targets `orbit` relation cannot be mapped to a source described with the `go-around` relation). Every candidate source retrieved by MAC/FAC must contain at least one predicate that is identical to one in the target probe. But "authors" of domain descriptions may differ in the terminology used to describe those domains, so the retrievability of a domain may rely on who generated the description. Furthermore creative analogies are often, if not

always, found between semantically distant domains (Boden, 1992; Hoffman, 1995). For example, physicists would generally represent the `attracts` relations in the solar-system and atom domains (Figure 2.1) as `gravitational-attraction` and the `electromagnetic-force` respectively (Figure 2.6). But MAC/FAC is blind to the indirect similarity between these descriptions.

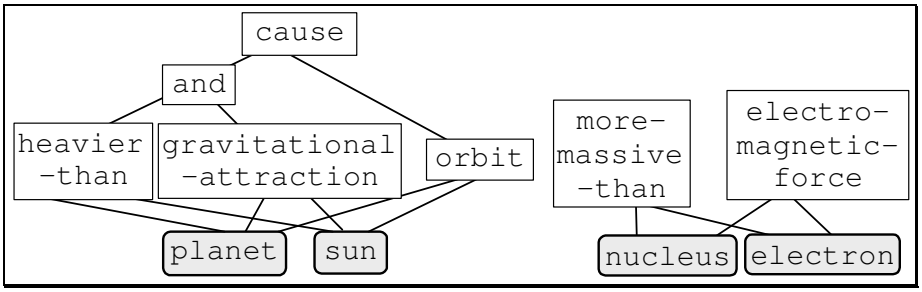

Figure 2. 6 - *Domains Before the Atom:Solar-system Analogy Was First Drawn*

In fact, identifying the `attracts` relations in both solar-system and atom domains, is really part of the *result* of the discovery of this analogy by Rutherford in 1911 (Miller, 1996; Wilson and Buffa, 1997). However, the objective in this thesis is to identify these analogies afresh - and not to benefit indirectly from their discovery by using the resultant terminology. So, in one sense, it could be argued that to date, mapping models have focused on elaborating known analogies, rather than inventing new analogies. This highlights the deep difference between access and retrieval made in Section 1.3.4. Generating a novel analogy typically requires *retrieving* a novel source domain, while interpreting a given analogy requires *access* to the given domains - whose description may highlight the relevant commonality (O'Donoghue and Crean, 2002).

SME is just as susceptible as MAC/FAC to variations in domain terminology[2]. If SME is given the more complex representation of the *solar-*

[2] The "Rerepresentation" (Yan, Forbus and Gentner, 2003) is not yet part of these models, and so it will not be discussed until a later Chapter.

system and *atom* domains, it will fail to find the mapping between `gravitational-attraction` and `electromagnetic-force`. Failure to generate the required mappings may also result in a failure to generate the required inferences - as it would in this instance. In Chapter 4 we will present retrieval and mapping algorithms that overcome these identicality-based problems, allowing the retrieval and mapping of semantically distant domains.

Phineas is the only model that addresses the verification process, performing deep reasoning using its elaborated problem representations. But its strength and weakness lies with these representations, which bind it closely to the domain of physical analogies. These extended domain descriptions are not generally available and this severely limits the generality of this approach. Notably, Phineas has not been routinely used in SME implementations.

Reeves and Weisberg (1994) divide the structure of a problem into three different levels of: the *problem context*, the *solution principle*, and the *final problem context.* Phineas performs retrieval by focusing on the *problem context* and the *solution principle*, while it performs validation by focusing primarily on the *solution principle* and *final problem contexts*. However, only the *problem context* will generally be known beforehand, and it is up to the analogy to identify an appropriate *solution principle* and the corresponding *final problem context*. We see all three levels as operating between predicates, which we refer to as the inter-predicate level. We will return to this inter-predicate (and intra-predicate) level in later chapters.

2.3 ACME and ARCS

Soon after the proposal of SME, an alternative model of analogy was proposed called ACME. ACME agreed with SME in its use of *structural constraints* to interpret a given analogy. However, it added *semantic* and *pragmatic* influences to the model. We begin by examining the ACME mapping model, which introduces many constructs that are also used in the associated retrieval model. Then we examine the ARCS retrieval model and the additional structures that it uses.

2.3.1 ACME

ACME - The Analogical Constraint Mapping Engine (Holyoak and Thagard, 1989) is a model of analogical mapping using a parallel constraint satisfaction network. The constraint network that underlies ACME is Grossberg's (1980) Interactive Activation and Competition (IAC) model of artificial neural networks (ANN). ACME assembles a custom-made ANN for each *source:target* problem, and it is this network that identifies the inter-domain mapping.

Figure 2.7 shows ACME's constraint satisfaction network for identifying the mapping in the *solar-system:atom* analogy. The inputs to ACME are the two domain descriptions. ACME begins by constructing a solution space of processing elements (nodes), each representing a single plausible match-hypothesis. Each node is labelled with a match-hypothesis (such as, *sun* = *nucleus*), though labels do not influence the networks operation. Positively weighted *excitatory* connections are inserted between structurally supportive match-hypothesis nodes, such as a predicate mapping supporting the mappings between the corresponding arguments. Conversely, negative *inhibitory* connections connect competing match-hypotheses, like alternate mappings for any given target element.

After construction, a small amount of activation is given to every node, which is a numeric value (a real value between 0 and 1) representing the current state of that processing element. The IAC *update rule* then computes new activation levels for each node. An *update rule* is a formula that calculates a new (numeric) activation value for each node, based on the activation levels of other nodes in the immediate neighbourhood of each node (Haykin, 1998). The IAC update rule allows activation to "flow" across positive connections, and avoids having activation at either end of an inhibitory connection. After a number of epochs (where all nodes in the network are updated), energy gathers in the largest collection of mutually consistent match-hypotheses. Simultaneously, energy leaves the unsupported match-hypothesis nodes leaving them relatively inactive. When the network converges to a stable state, the labels on active nodes (with activation above a predefined threshold) are read-off to reveal the inter-domain mapping.

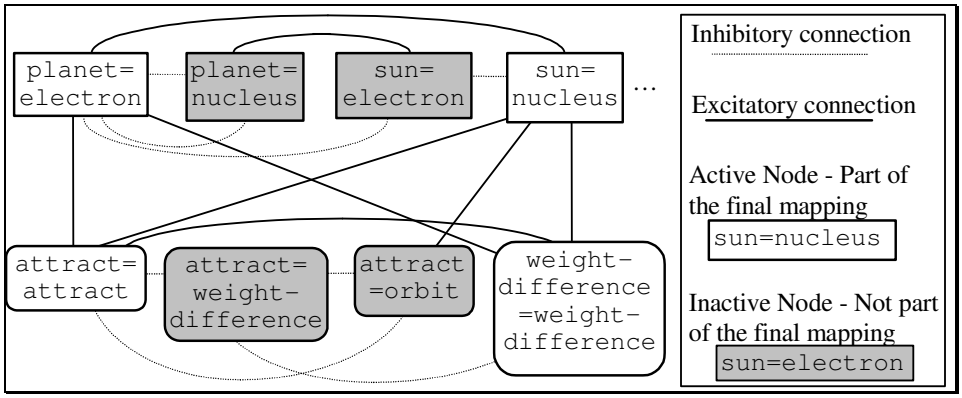

Figure 2. 7 - *A Simple ACME Solution Network*

One interesting feature of ACME is its ability to match non-identical predicates, via the *semantic similarity* constraint. This is achieved by using a semantic node connected to all semantically similar nodes, the strength of the connection being proportional to the degree of similarity between the matched items. In this way the final mapping is biased in favour of similar pairings, but this constraint is in direct competition with all other constraints represented within the network structure.

The *pragmatic centrality* constraint is supported in a similar manner. A particular match-hypothesis node that is known to be important, can also be included on the final mapping. To achieve this an excitatory link connects this node to an "input bias" to heighten its activity level. This sways the entire mapping to include this node, as well as nodes connected to it by excitatory links.

Thus, ACME can generate mappings by combining multiple influences (structural, semantic and pragmatic). However, ACME provides no guarantee about the resultant mapping. So, even a node that is indicated to be important may not be in the final mapping. ACME does not even guarantee to find the largest mapping, though tests (Holyoak and Thagard, 1989; O'Donoghue and Wyatt, 1995) show that the correct solution is generally found for smaller problem sizes (with less than 20 predicates in each domain).

ACME and Transfer

ACME does not perform inference in the traditional "pattern completion with generation" way. Any source information that does not have a mapping in the target domain must be treated in a special manner. ACME identifies two specific contexts in which the existence of (though not the identity of) transferable information may be known beforehand: *cross-structure queries* and *internal queries*. This partial information is used to inject additional "dummy" information into the target, so that it can map fully against the source. The source mappings of this dummy information are then used to identify the transferable material. However, we generally do not presume such dummy information is available before the mapping, but rather we derive these inferences from the mapping itself.

For example, in the solar-system example there are three source predicates without mappings in the target; `cause`, `and` and `orbit`. The atom target must include dummy information against which these relations can map. When nodes corresponding to this dummy information are found in the mapping, they are read off to reveal the transferable material. So ACME only generates inferences when suitable dummy information is included in the inter-domain mapping.

2.3.2 ARCS

ARCS - Analogue Retrieval by Constraint Satisfaction (Thagard *et al*, 1990) is a model of analogy retrieval and mapping that evolved from ACME. Like MAC/FAC it operates in two distinct stages: a retrieval stage and an ACME-based mapping stage.

Retrieval in ARCS uses the target domain as a *probe* into long-term memory, to identify domains using semantically similar predicates. ARCS uses the well known WordNet (Miller, 1991, Miller, 1995) lexical database to estimate semantic similarity. While ACME considers all possible mappings, ARCS uses WordNet to consider only semantically similar mappings. This helps reduce the number of match-nodes and links that are created for each source domain, and increases the range of sources considered.

So for the solar-system:atom example, the `attract` predicate in the target will retrieve all sources containing `attract` and synonyms of `attract`. WordNet 1.7.1 identifies three synsets for `attract` encompassing the terms: `pull`, `pull in`, `draw`, `draw in`, and `appeal`. Different WordNet relations correspond to different link values (*eg* synonym=1.0, super-ordinate = 0.3 and antonym = -0.4) with varying degrees of impact on the mapping identified. Each mapping unit is linked back to an external semantic unit, with a connection-strength equal to the semantic similarity between them. It should be pointed out that (like ACME) ARCS is not sensitive to the particular values used on these links, although antonyms must have a negative weight.

In the mapping phase, the target along with each candidate source is used to build a very large constraint satisfaction network. Again each node represents a match-hypothesis between a target element and a corresponding element from one of the source domains. Links represent mutual support and competition between these mapping nodes. Because the network contains multiple source domains, the match-hypothesis nodes from alternate sources are mutually competitive. Additional match-hypothesis nodes are constructed to represent the mapping between the domain names themselves. So, one node might represent the atom:solar-system analogy, while another might represent the atom:water-flow analogy. These nodes are connected by excitatory links to the individual match-hypothesis nodes representing that mapping. So, ARCS performs semantically similar domain retrieval, and identifies one candidate source through a more detail mapping process.

2.3.3 Evaluation of ACME and ARCS

ACME extended the range of influences on the mapping phase, incorporating structural, semantic and pragmatic factors within a parallel constraint satisfaction network. It also demonstrated that these multiple influences can operate in parallel, without any one constraint dominating all others. However, there are three main criticisms of ACME. First, there is the scalability problems of ACME. Consider some results taken from (O'Donoghue and Wyatt, 1995; Veale, Smyth, O'Donoghue and Keane,

1996), and summarised in Table 2.6. This table summarises a number of mapping experiments conducted on ACME, using domains ranging in size from 6 predicates to 26 predicates. (Larger domains caused the system to crash due to the extreme memory requirements).

	Number of Predicates	Number of Nodes	Number of links	Solution Times(seconds)
Experiment 1	6	62	314	40
Experiment 2	9	135	1052	225
Experiment 3	11	223	3608	570
Experiment 4	16	583	10183	2430
Experiment 5	20	786	16000	4500
Experiment 6	24	1094	27535	20000
Experiment 7	26	1252	32443	60000

Table 2. 6 – *Growth in ACME Parameters with Problem Size*

As Table 2.6 indicates, the number of nodes in the generated network grows polynomially with increasing problem size. However, this growth is dwarfed by the exponential grown in the number of links that are inserted between nodes within the network. These increases in the number of nodes and links required by an ACME network, causes an exponential growth in the amount of time required to even construct the ACME mapping network; that is, just to represent the analogy (O'Donoghue and Wyatt, 1995). The solution time required for the network to actually reach a converged state (and provide a solution) also grows exponentially with problem size - and this effectively limits ACME to solving smaller mapping problems. The time required to reach convergence is indicated by the thick line in Figure 2.8 (*not* including network construction time). Holyoak *et al* (1989) do indicate that convergence time is roughly constant in the number of epochs required,

however the amount of time required for each of these epochs increases with the number of nodes in the network (O'Donoghue and Wyatt, 1995; Veale *et al*, 1996). Furthermore, ACME is programmed to cease after 100 epochs, and has generally not reached convergence in such a time for the larger problems. Performance is therefore a severely limiting factor for ACME on larger problems.

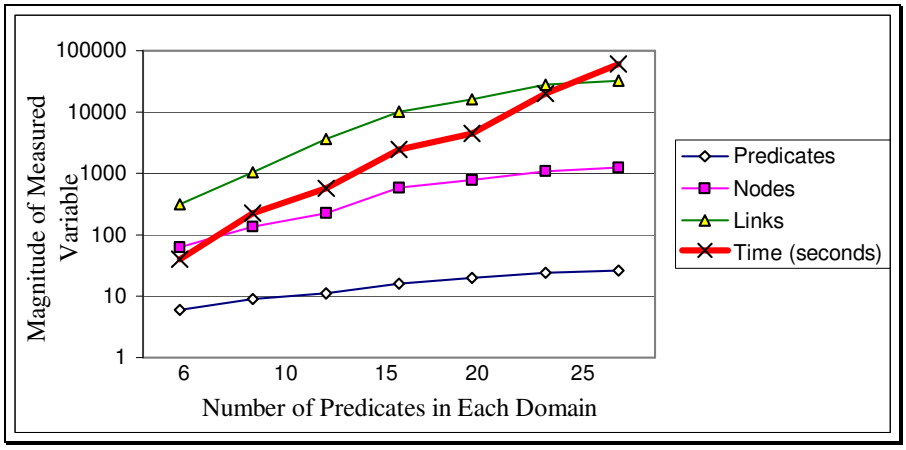

Figure 2. 8 - *ACME network Grows Exponentially with Problem Size*

The second criticism of ACME relates to its ability to correctly identify in the correct inter-domain mapping. These same experiments also revealed that ACME will not always generate a useful mapping. The underlying IAC neural network model (Grossberg, 1978) is not guaranteed to ever reach convergence, unlike say a Hopfield network (Wasserman and Oetzel, 1990; Haykin, 1998). Hopfield networks for the Travelling Salesman's Problem (another NP-Complete problem) frequently fail on 20 city problems - and rarely succeed on 25 city problems (Wasserman and Oetzel, 1990). So, a converged state may not necessarily correspond to a viable inter-domain mapping. Furthermore, the network may not even converge, but oscillate continuously between a number of alternate states. A related issue is that partial converge of the network does not reveal a partial answer, so there are no obvious short-cuts around these performance problems.

The final criticism of ACME centres on its treatment of inferences. ACME does not perform transfer in the traditional "pattern completion with generation" (Holyoak and Novick, 1994) understanding of this term, neither does it validate these inferences. So, while ACME may be capable of mapping creative analogies, it does not generate or validate the inferences in an effective manner. In conclusion, ACME is an interesting model of mapping, but it suffers from severe scalability problems. It does, however, allow structural, semantic and pragmatic forces to influence the mapping process.

ARCS Discussion

The problems associated with the ACME network also affect ARCS, which is more complex in a number of respects. ARCS generates much larger networks involving multiple sources, creating even larger networks. These networks will be even slower to generate and to reach convergence. These larger networks are even less likely to reach a useful final state (just as the larger Hopfield network rarely produce a viable solution to the Travelling Salesman's problem).

One major limitation of ARCS is that it is designed to identify only a single best source, from among the candidate sources. But the unpredictable (and even *improbable*, Boden, 1998) nature of creativity, typically means we must explore a large number of candidate sources before a useful source is found. Even re-ordering the "losing" domain-level nodes according to activation level would not necessarily correspond to their potential as alternate sources. To identify the second best source, ARCS must remove all nodes related to the winning source and re-start the convergence process. Thus, ARCS is not well suited to acting as the retrieval engine for a creativity model (Boden, 1994).

ARCS should be praised for its use of WordNet to support semantically flexible retrieval. However, its reliance upon the same underlying mechanisms as ACME condemns ARCS from a purely computational perspective.

2.4 IAM

Following the combined mapping and retrieval models of SME and ACME, IAM is a pure mapping model. The Incremental Analogy Machine (IAM, Keane and Brayshaw, 1988; Keane *et al*, 1994) is a computational theory of analogy's mapping phase. IAM attempts to explain mapping generation in a cognitively plausible manner, effectively taking account of the influence of working memory limitations on the mapping process. This model explains why increasing the number of similar elements in two domains can, under certain circumstances, decrease the amount of time required to identify the correspondence. It also helps explain and predict why humans do not always generate the largest possible mapping.

Not only is IAM a more cognitively plausible model of mapping (Keane 1997), it is also more computationally efficient than its predecessors (Keane, Ledgeway and Duff, 1994). When presented with two (possibly incomplete) domains, IAM first tries to identify a *seed group* of predicates from each domain from which to grow the final mapping. The selection of this seed group is the key to the whole IAM approach to mapping. A group is any set of connected or systematic predicates in a domain, where this connectivity is represented by argument structure. For example in the *atom* domain, one group would include the `weight-difference` relation and its two arguments, and in the *solar-system* domain a group would contain the `cause` relation, its arguments, and their arguments' arguments in turn. Effectively, a single group represents the entire *solar-system* domain (see Figure 2.9). Predicates and objects are often members of multiple groups, like the `nucleus` belongs to both the `weight-difference` group and the `attracts` group.

IAM then ranks all groups so that the groups with the most high-order systematic structure receive the highest rank. High-ranking groups from each domain are selected to form the initial (seed) mapping between domains. This maps all elements within the two groups (within the bounds of the normal 1-to-1 constraint). This initial seed-mapping forms the basis upon which the remainder of the mapping is compiled. The unmapped seed groups are

selected in turn, and added to the inter-domain mapping where possible. This process continues until there are no more unmapped seed groups, or no other seed groups are compatible with the mapping.

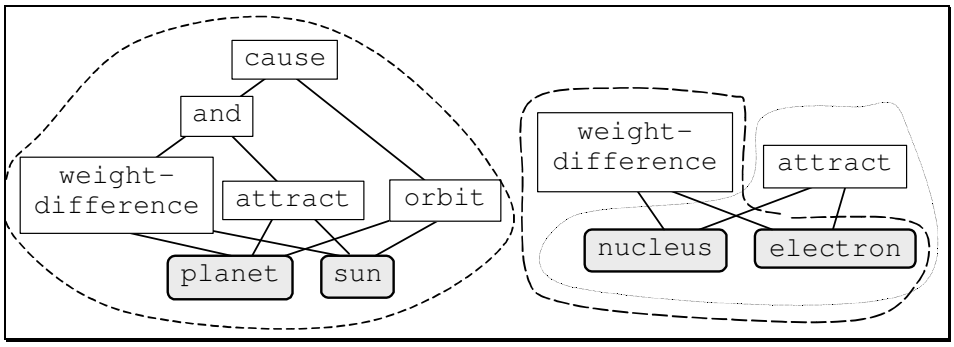

Figure 2. 9 - *Seed-Groups from the Solar-system and Atom Domains*

As seed elements are matched alternative matches are noted, so that an alternate mapping can be generated if the original fails to yield sufficient matches. IAM uses serial constraint satisfaction to distinguish between useful and undesirable matches, based on pragmatic, similarity and structural factors. It should be noted that most groups are quite small and even large domains are (generally) composed of interconnected sets of smaller groups. As seed elements are matched alternative matches are noted, to create an alternate mapping should the original fail to yield sufficient matches.

Learning is achieved by transferring unmatched source items to the target domain. One particularly interesting feature is IAM's ability to incrementally generate inferences. So any inference from an earlier mapping forms part of the mapping, and can thereby support later mapping and inference processes. This transfer process is followed by an evaluation of the mapping, to ensure that at least half the elements in the seed group have been mapped. If less than half are mapped, then a re-mapping is performed beginning with alternative seed mappings and should that fail alternative group mappings.

2.4.1 Evaluation of IAM

IAM's main significance is that it attempts to be a cognitively plausible model of mapping. As such, it highlights the performance difficulties of O/SME (Falkenhainer, Forbus and Gentner, 1989) and G/SME (Forbus and Oblinger, 1990) and ACME (Holyoak and Thagard, 1989). The efficiency of the IAM algorithm can be greatly attributed to the identification of root predicates within a domain. Its efficiency can be illustrated best by considering its operation on an isomorphic domain-pair. Well understood domains like the *solar-system* domain in Figure 2.9 may contain a large number of predicates that are controlled by just a few causal root-predicates. Originating the mapping process in these root-predicates vastly reduces the search space that is considered - and thus reduces the computational complexity of the mapping problem. Thus, cognitive plausibility and computational efficiency seem to mesh well in this algorithm.

IAM does not address the spontaneous *retrieval* of sources, for some presented target. However, it may be argued that its incremental nature supports *access* of an indicated source domain (the distinction between *retrieval* and *access* was outlined in Section 1.3.4). That is, once the source domain for a given target is known, IAM addresses how information may be brought from memory and incorporated into the mapping. Thus, it is more amenable to problem-domain "access" than the earlier mapping models would be without a distinct *retrieval* model.

The incremental nature of IAM also lends itself to identifying correspondences within a memory-embedded environment. That is, if memory is organised as one large inter-connected network of concepts, IAM might generate a seed mapping given two starting nodes. If this initial mapping is successful, expanding the source and target nodes might serve to incrementally develop the mapping. This makes IAM a very flexible and adaptable mapping model.

We see this incremental quality of the mapping phase as one which could help resolve the apparent conflict between the exponentially increasing expense of the mapping task, and people's ability to develop and utilise very large analogies - seemingly without domain boundaries. Complex analogies

are often built dynamically, with extra information being added to the comparison once the initial mapping is formed. This may involve accessing information that is indirectly related to the domain in question. If inappropriate information is retrieved, validation might reject the corresponding inferences until the correct extension of the analogy is identified. Incremental mapping interacts naturally with memory, accessing subsequent parcels of information and adding them to the mapping as appropriate.

2.5 Lisa

The earlier models addressed the mapping process, but did not explicitly highlight the connection between mapping and the induction of more general schemata (Gick and Holyoak, 1983). Among other things, Learning and Inference with Schemas and Analogies - Lisa (Hummel and Holyoak, 1996; 1997; Holyoak and Hummel 1998) attempts to model this association between analogies and schema induction. Lisa is based upon Shastri and Ajjanagadde's (1993) model of synchronic activation for temporal binding. This neural network model uses temporally synchronised activation between neurons, to signify a binding between a *variable* and its *value* within a connectionist framework. Lisa uses this temporal synchrony to signify a mapping between source and target elements. Although it is focused on mapping and induction, Lisa also addresses retrieval by using the binding both to identify and to represent the inter-domain mapping.

A reduced version of the *solar-system:atom* analogy, as may be represented by Lisa, is illustrated in Figure 2.10. The key to understanding this diagram is that a predicate combines the arguments it uses (`sun`), together with the argument-position it plays in relation to that predicate. So, if the `sun` is used as the first argument (ie the `patient` role), this will be treated differently to using this argument as the second argument (ie `patient` role).

Object nodes represent attributes such as `heavy`, `massive` and `hot`, as might be linked to the `sun` object. Semantic nodes also represent role information; thus the agent and patient arguments of the `weight-`

`difference` predicate are represented by connections to semantic units representing those roles (see the bottom layer of Figure 2.10). So one role may represent the `weight-difference` agent role (`w-d-1`), while another represents the `weight-difference` patient role (`w-d-2`). A *sub-proposition* layer binds the objects to the roles they play, so the agent role of the `weight-difference` predicate and the `sun` are bound together by the `w-d-sun-1` node, while `w-d-planet-2` represents the patient argument binding. The *Predicate and Object* layer represent both the predicate and objects units that are contained within the domains. All analogs are represented in a similar fashion, so the `nucleus` concept in the *atom* domain might be linked to a few of the same semantic primitives.

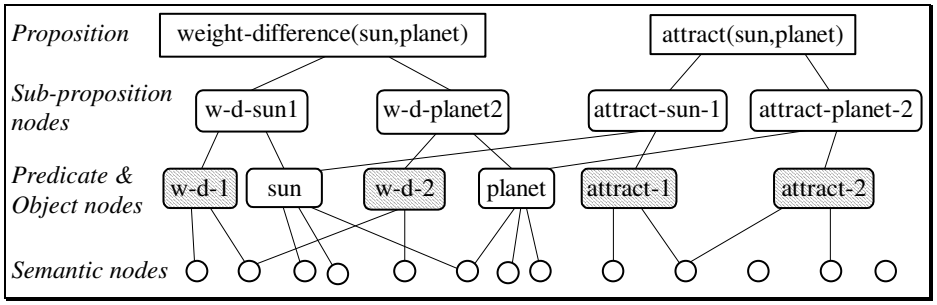

Figure 2. 10 - *Lisa's memory structure*

2.5.1 Retrieval and Mapping in Lisa

Lisa focuses on mapping and induction, however its design naturally extends to support a model of analogy retrieval. Crucial to all phases of Lisa's operation are the different memory structures that it uses. Lisa uses a number of different types of memory; *active memory* identifies domains that Lisa is currently evaluating, while *LTM* (Long Term Memory) representing all other domains. Lisa also makes extensive use of *semantic primitives* to represent the association between semantically similar concepts (predicates, objects, attributes etc.), where similarity is represented by connectivity to these shared semantic primitives.

Retrieval is supported in Lisa by directing activation from the target's (or *driver*'s) top-level propositions, activating in turn the *proposition* units

(see Figure 2.10), *sub-proposition* units and "*Predicate and Object nodes*" in LTM. These activate a variety of semantic primitives associated with each concept. These activated semantic primitives respond by activating further *sub-proposition* units that then activate their corresponding *proposition* units. Thus activation reaches all *proposition* units that are semantically similar to the target, and these active *proposition* nodes identify the set of candidate sources. So for the *solar-system:atom* example, activating the `weight-difference` relation in the *atom* target activates the *sub-proposition* nodes (`w-d-nucleus-1` etc), *Predicate and Object* nodes (`w-d-1`, `nucleus` etc), and semantic nodes (`sub-atomic`, `large` etc). These nodes activate all connected nodes, including the `weight-difference` relation from the *solar-system* domain. The role nodes (`w-d-1`, `w-d-2`) are also activated in the *solar-system*, along with the objects that share similar attributes (`nucleus`). Thus, the *solar-system* source may be identified from memory.

A precursor to mapping in Lisa is the existence of dormant 1-to-1 *mapping connections* between potential mapping units in the different domains. (Indeed, these mapping connections are the essential difference between retrieval and mapping). Mapping connections exist between structure units of the same level – so all sub-proposition units in one domain share a connection to all sub-proposition units in the other domain, and so on for the other unit types. The purpose of mapping then, is to identify which of those connections form part of the current inter-domain mapping.

Mapping connections are active entities that identify the presence of simultaneous activation at either end of their connections. Each connection contains a temporary buffer to accumulate evidence that connected elements participate in the final mapping. Synchronic activation is then spread from the source and target *proposition* units, and then onto the *sub-proposition* and *semantic* units, with all activation affecting the corresponding mapping connections. Occasionally the permanent weights are updated according to the strength of the temporary buffers, and the buffers are then flushed. This generates large positive weight values upon mapping connections and large negative weights on the non-mapping connections.

The synchronic activation that is spread from the source and target propositions, generates activation across the two domains. Some nodes will be activated in synchrony, and these are referred to as the *phase set*. The mapping process effectively identifies nodes that have entered a state of "phase locked activity". So for the *solar-system:atom* analogy, the two `weight-difference` propositions enter the phase set and contribute to the mapping. Their agent roles are also activated synchronously, and so the `nucleus` and `sun` enter the phase set - as are any attributes they share (eg `large`). Thus, the mapping is built up as parallel portions of the two domains enter the phase set.

2.5.2 Schema Induction in Lisa

Lisa views the induction of schemata as an integral part of analogical reasoning, occurring in parallel with the mapping process. While the *phase set* is evolving from synchronic activity on the mapping connections, Lisa induces schema nodes based on the semantic activity occurring across semantic nodes. The first class of schema induction that is supported is *predicate-centred induction*, and this creates general schemata from specific instances. Lisa induces schemata whenever "sufficiently novel" analogies have been found, recruiting new *proposition, sub-proposition* and *object* units as required to instantiate the schema. As Lisa uses the semantic commonality as a basis for this induction, semantic units that are common to both domains form the content of the new schema. To ensure that only units common to both domains form part of the schema, Lisa uses an activation function that strongly identifies units on both streams of activation. This technique of intersection discovery operating on the *solar-system:atom* analogy might generate a schema representing "a heavy thing that attracts a lighter thing, may cause the lighter thing to orbit around it". This schema exists as a separate entity and effectively becomes independent of the inter-domain mapping.

The second class of induction is object centred schema induction, or *compressed mode* operation in Lisa parlance. This is essentially a form of predicate discovery. Two predicates are compressed only when they refer to

the same object. In this mode, it is the objects that control the activation (not predicates), so that the following mapping:

```
weight-difference (nucleus, electron)
attracts          (nucleus, electron)
```

induces a schema `weight-difference-attracts(nucleus, electron)`, compressing two predicates into one. As noted by Hummel and Holyoak, this compressed predicate can be represented in a *phase set* of smaller size than when represented by multiple predicates. This is because of the reduced number of roles that are involved in this *compressed* predicate.

2.5.3 Transfer in Lisa

Lisa also supports analogical transfer as a form of unsupervised learning induced by the source in the target domain. The mapping is first identified by using the target as a driver, to identify the mapping connections between the two domains. Then Lisa uses the source as the driver, but keeps the previous mapping connections. This induces a new pattern of activity in the target domain, because of source information that has no counterpart in the target. This new pattern of activity at the target end of the mapping connections (and beyond the phase set) causes the unsupervised learning algorithm to recruit new nodes in the target domain. These new nodes are connected to those source items for which there was no counterpart in the target - thereby effectively performing transfer. Although Lisa may not generate a new label for these entities, their semantic information will be correct according to the mapping.

2.5.4 Evaluation of Lisa

Lisa highlights the close relationship between analogy and the induction of generalised schemata. Lisa describes how an analogy can produce a mapping and a schema simultaneously. It makes interesting use of a model of artificial neural networks based on temporal synchrony, and applies it analogy retrieval and mapping. Synchronic activation is also used for schema induction and for discovering new *compressed* predicates.

The authors acknowledge two significant limitations of Lisa, related to its use of dynamic binding. The first is a capacity limit that constrains the

number of individual dynamic bindings that can be represented. The capacity limit on synchronic binding is equivalent to the size of the *phase set*, which has been set to a maximum of six in Lisa. (This is neurologically justified from studies carried out on the *visual* systems of cats and monkeys, and is also apparent from human studies). Thus, Lisa can generate mappings between at most six nodes for any given analogy. This prohibits Lisa from mapping any pair of predicates that are connected to a large number of semantic or other units. It should be pointed out that Lisa could generate mappings between larger domains if the self-imposed (and neurologically justified) constraint of a small phase set size were removed.

The second restriction is the one-level restriction, whereby dynamic bindings can only represent one level of abstraction (or hierarchy) at a time. While Lisa deals adequately with the co-ordination amongst mapping units below the *proposition* level, co-ordination between multiple *proposition* units is by no means straightforward. The distributed (*ie* non-centralised) nature of synchronic activation is difficult to control, and a principled way to map large numbers of inter-domain correspondences seems unclear.

> *"As a default, propositions are selected* ***one*** *at a time to become active in the phase set. … The protocol for specifying the order in which propositions are selected is given as input."* (Hummel and Holyoak, 1997)

Mappings that are established by one proposition are stored as weights that bias the subsequent mappings - this ensures that the 1-to-1 constraint is not violated. Of course, the order in which these propositions are presented might also affect this biasing mechanism. Lisa has still not demonstrated that it can map large numbers of predicates. Lisa therefore has no means of supporting the systematicity principle, and cannot identify large useful mappings. Perhaps Lisa could form part of a larger mapping model, using the incremental mapping approach (Keane and Brayshaw, 1988; Keane, 1990) to iteratively generate large mappings.

A novel feature of Lisa is its ability to break the "n-ary restriction" in mapping. This is best described by example, so consider two instances of the `larger-than` relation; `larger-than(a,b)` and `larger-`

`than(b,c)`. Lisa can generate a ternary predicate encompassing these two binary predicates; `larger-than(a,b,c)`.

The retrieval model used by Lisa is semantically based, though the use of semantic primitives' means that it is not limited to identicality based retrieval. Its use of distinct case-role nodes also allows domain topology to influence retrieval. However, retrieval is primarily a semantically based activity, which makes semantically distant sources difficult to identify.

Lisa's mapping model focuses on mapping pairs of propositions, not pairs of domains. While these may be semantically dissimilar propositions, Lisa still needs an external process to identify which pairs of propositions should enter the mapping stage simultaneously. This becomes even more of a problem with creative analogies, when we cannot rely on predicate identicality to help select mapping propositions.

While Lisa's approach to transfer is innovative, it may be problematic as it does not generate an identifiable label for newly generated target items. However, enough information exists in the transferred items that further processing may resolve this problem. Finally, Lisa does not address the validation process. so the generated inferences are not tested for validity. So, while Lisa is an interesting model of analogy at the fine-grained level of individual propositions, it does not support analogies between large novel domains.

2.6 ASTRA

Astra (Eskridge, 1994) is a multi-phase model of analogy that focuses on retrieval and mapping, but also addresses the "transfer and use" phase. Rather than focusing on individual stages, Astra shifts the focus onto the interactions between phases. Eskridge argues that isolated phases of analogy are easier to model than multiple-phases. Multi-phase models must address the communication problems that occur between phases, while individual phase models can rely on simpler *"atheoretical mechanisms"* to cover absent segments. Thus, many individual-phase models do not easily integrate into a larger multi-phase model.

Astra identifies memory as a central element within the analogy process. Significantly, all of Astra's processes communicate directly with a memory that is accessible to all phases (see Figure 2.11). Phases communicate with each other through this memory, using spreading activation as the common access method. This communication allows concepts that were weakly primed by one process (typically retrieval) to communicate indirectly with other processes (*eg* mapping and the transfer and use). Each process can add to the activation of connected concepts, possibly causing concepts to become the root of further processing. So for the *tumour:fortress* analogy, identifying that the `X-rays` can be split into `beams`, means that retrieval can search to see if the `army` can in turn, be split into smaller units.

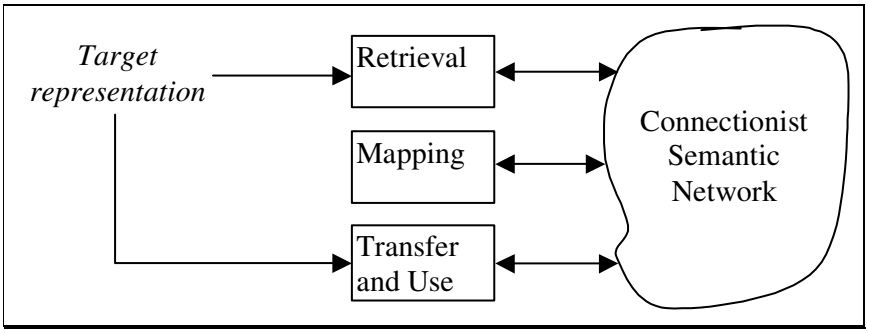

Figure 2. 11 - ***Architecture of Astra***

Astra starts with the presentation of a target, which instigates the retrieval process. Retrieval uses a joint spreading activation and marker-passing algorithm that both searches the knowledge-base for suitable sources and attempts to elaborate the target description by pattern completion. When goal and context information are present in the target description, this is activated and used as the origin for spreading activation by the "transfer and use" phase. (Thus, the transfer and use process can also be involved in retrieval).

The heart of Astra's mapping model centres on "conceptual bridges" connecting items in the source and target domains. Astra automatically inserts new conceptual bridges between corresponding concepts in each domain,

allowing activation to flow between related concepts. Bridge creation is invoked whenever activation from a node in one domain reaches a node in the other domain. Bridge creation is subject to two Bridge Creation Rules (BCR's). The first BCR rule states that if a pre-existing bridge exists between concepts, then increase the strength of that bridge along with any subordinate bridges (this is really a bridge strengthening rule).

The second BCR rule creates new bridges when activation reaches a new node and no previous bridge exists. However before inserting the bridge, this rule first compares the case-relations at either end of the bridge and if they match then a bridge is created (otherwise, bridge creation fails). This highly directed manner of inserting analogical bridges implements the structural consistency rule. Increasing the activation on mutually supportive bridges enforces the systematicity constraint.

Partial mappings can be extended by identifying other source nodes not yet included in the mapping, and determining their applicability within the mapping. To avoid overwhelming the mapping process, this mapping extension process only acts on the most highly active candidate sources.

Astra relies on four Spreading Activation Rules (SAR) and two Marker Passing Rules (MPR). SAR-1 dictates that activation be spread upward from an instance to a class node without decay, allowing all other instances of that concept to become active. SAR-2 increases the activation of nodes that are referenced as part of the goals of a problem, and SAR-3 increases the activation level of nodes with the same label as the origin of the spreading activation. Only SAR-4 allows the standard activation decay, weakly activating distant concepts. MPR-1 enables markers to be passed between nodes, and MPR-2 supports the creation of Bridge links under certain circumstances.

Transferable items are identified as source elements that have received activation, but do not have an associated bridge. These items are transferred to the target along with any relevant structure.

2.6.1 Evaluation of Astra

Astra's use of a structured localist-connectionist memory combined with spreading activation is a very useful idea. Allowing inter-phase communication through this shared memory helps keep each phase de-coupled from the other phases. However, Astra's spreading activation mechanism is an extremely expensive one, and its expense increases with memory size. In a large memory SAR-1 (spreading activation up a hierarchy) could potentially activate a significant portion of memory without any decay in activation strength. BCR-1 and BCR-2 mean that bridges can only be generated between semantically similar nodes. This eliminates the ability to identify truly novel comparisons, such as between objects that are used differently in two both domains. (A fish can be like a baseball-bat if it hits you!). This must be seen as a severe limitation of the Astra model. So, even though much of memory is being activated, it is still too semantically constrained to demonstrate the sort of novelty we want to see in our model.

2.7 SAPPER

Sapper (Veale, 1995) is broadly similar to the Astra model, but focuses on the efficient development of large inter-domain mappings. Among Sapper's strengths are it's ability to identify, in an efficient manner, both "object centred" and "predicate centred" mappings (which we describe below). Sapper is based on a joint localist-connectionist and symbolic architecture. Sapper allows the problem domain information to be integrated into a structured background memory, and Sapper operates on this memory to identify the inter-domain mapping.

Sapper operates in two phases; a preparation phase and a problem solving phase. The preparation phase adds structures to memory to expedite the subsequent mapping process. Sapper employs two rules to insert additional dormant bridges into memory, called the *triangulation* and *squaring rules* respectively (Figure 2.12). Like the earlier Astra model, these dormant bridges represent the possibility that the paired elements form part of some inter-domain mapping. During the second problem solving phase where a given analogy is interpreted, some of these dormant bridges are

"awakened" (as with the Astra model). Sapper's focus is on identifying which of the many existing inter-domain bridges should be awakened by a particular analogy.

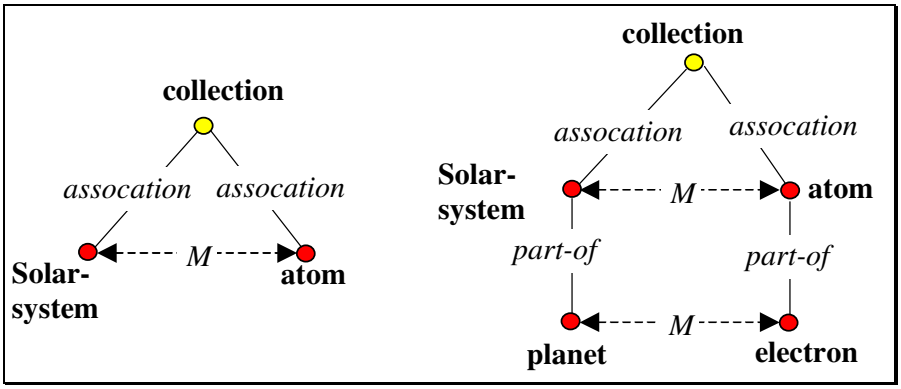

Figure 2. 12 - *Sapper's Squaring and Triangulation Rules*

Two bridge creation rules are central to the Sapper algorithm. These rules annotate memory with metaphoric links (M-links), indicating that the paired elements might form part of some inter-domain mapping. The *triangulation rule* inserts an M-link whenever two concepts share an association with a third concept (Figure 2.13). M-links inserted by the triangulation rule are based on the inherent semantic similarity between the two concepts. The *squaring rule* is based partly upon the M-links laid down by the triangulation rule. The squaring rule adds additional M-links between objects that are connected by the same relation to M-linked objects. Frequently, these concepts will be semantically different - overcoming the triangulation rule's semantic restriction. So the *triangulation rule* will insert an M-link between the `solar-system` and `atom` objects and between the `planet` and `electron` (Figure 2.13). The M-link between the `sun` and the `nucleus` is inserted by the *squaring rule*, as these nodes do not share any direct super-ordinate. However, both the `sun` and `nucleus` are linked by the `part` relationship to M-linked nodes (see Figure 2.13). Thus, mapping possibilities are propagated across many diverse concepts, and Sapper is ready to generate analogical mappings.

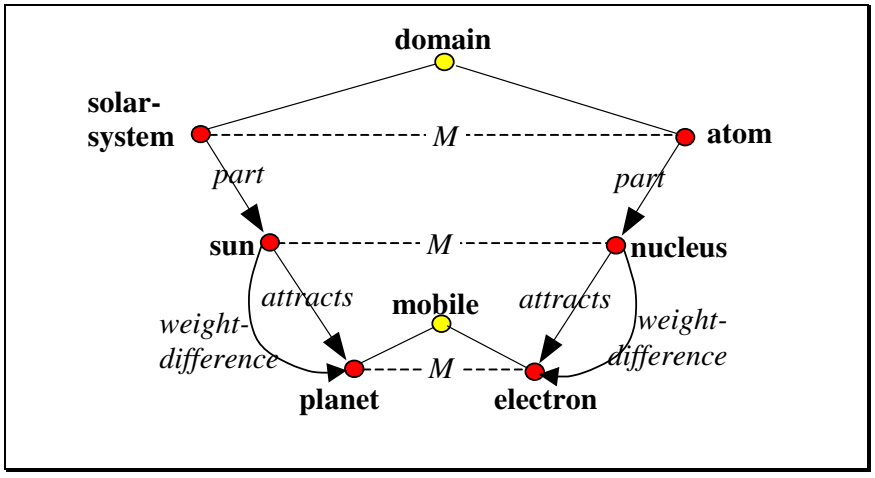

Figure 2. 13 - *Some "M-links" for the solar-system:atom analogy*

Mapping commences by spreading activation simultaneously from the source and target nodes, iteratively activating connected nodes. Both source and target activation streams also contain different markers. When two different markers are detected at either end of an M-link, the corresponding items are added to the inter-domain mapping and spreading activation continues. In this way analogical comparisons are built-up, gradually spreading across memory. However, activation decays across successive links, and items beyond the activation horizon are not added to the mapping.

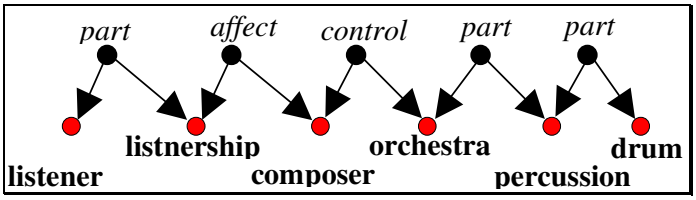

Figure 2. 14 - *Predicate-centred domain representation*

So for our *solar-system:atom* example, the `sun` and `nucleus` nodes will be activated and will be mapped together because of their connecting bridge (inserted by the squaring rule). Nodes adjacent to these will also be mapped, provided they are also connected by M-links. So, the `planet` and the `electron` will be added to the mapping. Thus the two `attracts` relations will also be mapped to one another. This process continues until all

domain elements are mapped, and no more items are available be added to the mapping.

Consider the data represented in Figure 2.14 (from Veale *et al*, 1995) where data is represented in a lateral manner. This predicate centred view is favoured by models like SME (Forbus, Ferguson and Gentner, 1994). However, Sapper's use of spreading activation utilizes the inherent hierarchical structure shown in Figure 2.15 (note that this is a different presentation of the same predicates as displayed in Figure 2.14). Sapper's memory model identifies and uses the inherent hierarchical structure, regardless of how the problem is described. This attractive quality enables the efficient and flexible identification of mappings for domains described in different ways.

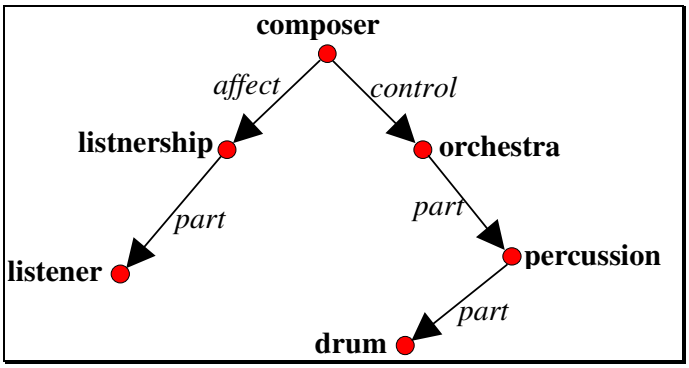

Figure 2. 15 - *Object centred domain representation*

2.7.1 Evaluation of Sapper

Sapper is intended as a model of metaphor interpretation (Veale, 1995), and so it does not directly address retrieval of source domains, inference or adaptation. It does however address source domain access, and can generate large mappings extremely quickly (see Veale *et al*, 1995, 1996). Mapping takes place within a large integrated long-term memory, with all source, target and background information stored in the same homogenous store. (Unlike other models that use isolated domain descriptions). Thus, analogies are extracted (or accessed) from an integrated memory structure using a spreading activation mechanism. Spreading activation allows object-centred

and predicate-centred mappings to be identified with equal efficiency. Sapper's main advantage over other models is its computational efficiency and speed, allowing large and unbounded metaphors to be compiled.

Although the prior inclusion of M-link bridges in memory may seem to contradict the claim that analogies generate new similarities, Sapper is capable of adding these at run-time - though with a mild performance penalty. The number of M-links required by Sapper appears to grow roughly linearly in memory size (Veale *et al*, 1996). However within a very large memory, this may prove unwelcome.

Sapper appears amenable to encompassing additional phases like transfer and adaptation though this has not been done. It is also significant that the mapping process is performed within a memory embedded environment. This could allow information from the mapping process to interact with retrieval, transfer and validation, perhaps by varying the activation levels on the relevant nodes.

2.8 AMBR - Associative Memory Based Reasoning

Kokinov sees analogy as part of a reasoning continuum, ranging from deduction through analogy to induction. So the AMBR (Kokinov, 1994, 1998; Kokinov and Petrov, 2000-a, 2000-b) model can be considered a (somewhat) general cognitive architecture (Anderson, 1983; 1993), capable of modeling many different cognitive processes. AMBR attempts to capture some of the generality of memory-based reasoning in a model of analogy that is composed of communicating parallel processes.

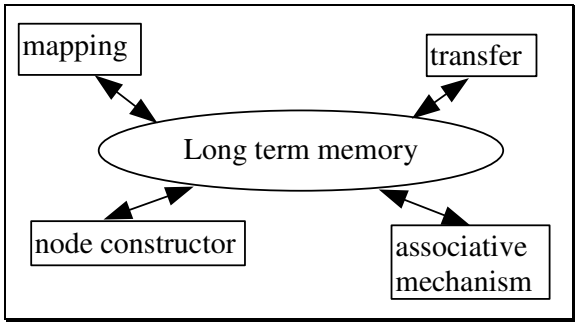

Figure 2. 16 – *Architecture of the AMBR Model*

Kokinov identifies five phases of analogy, namely*; retrieval, mapping, transfer, evaluation* and *learning*. Each of these five phases executes in parallel, and communicates with other phases through the shared memory (see Figure 2.16, adapted from Kokinov, 1994). AMBR focuses primarily upon the *retrieval* and *mapping* phases, and while the first four phases are broadly similar to those of other multi-phase models of analogy, the last phase is different. This *learning* phase modifies the entire reasoning process, to improve its later problem-solving ability.

AMBR's memory is a joint localist-connectionist network, where world knowledge is represented by frame-like descriptions stored at nodes in memory. Activation level corresponds to the degree of relevance of that node to the current situation. Nodes themselves are responsible for calculating activation values, and for propagating activation between connected concepts. Links then, have symbolic labels that connect concepts and allow activation to reach indirectly referenced concepts.

AMBR supports both automatic and strategic retrieval using spreading activation. Automatic retrieval is based on setting goal lists and priming the appropriate concepts in memory to activate the desired source. This strongly favours semantic locality in retrieval, unless the goal nodes are explicitly identified to force activation onto a semantically distant source. Strategic retrieval begins by spreading activation at the target and an identified source. Retrieval activates nodes in LTM that will be picked up by the subsequent mapping process.

Mapping in AMBR is quite like ACME (Holyoak and Thagard, 1989), but the constraint satisfaction network is composed "in situ" in LTM by inserting additional nodes and links (as opposed to ACME's separate mapping space). AMBR generates *correspondence nodes* (ie mapping nodes) based on the semantic similarity between the mapping elements. Additional correspondence nodes are then created by exploiting the structural similarity between the two domains. Correspondence nodes are inter-linked by temporary excitatory and inhibitory connections that implement structural, semantic and pragmatic constraints. The initial activation level of each

correspondence node is derived from the semantic similarity of the constituent items. Applying an iterative update rule and allowing the network to reach convergence highlights the required mapping nodes that constitute the inter-domain mapping.

AMBR supports three different classes of transfer. The simplest is when the target structure is composed only of mapped elements, so inference is achieved by pattern completion. Secondly, when a target relation is missing (so only objects are mapped), the source predicate or some appropriate super-class of this predicate, is added to the target. Choosing between the available super-classes is based on the activation level of the available nodes. Finally, where only the predicate correspondence is known (and no objects are mapped), a new target object is constructed that satisfies all argument restrictions found in the target description. Also, AMBR performs some predicate adaptation based upon known information and previous examples of the relevant predicate.

Transferring a predicate in AMBR tries to use existing target objects of the same classes as the source objects, using these objects to construct the new predicate if available. (So for the tumour:fortress example, if the target does not contain the `beam` object, then AMBR will create a sub-class of `platoon` in the tumour domain) Otherwise, AMBR creates new instances of the corresponding objects (using prototypes), using these to construct the analogical inference. Now we describe an example of predicate transfer in the following analogy:

`in(water, container) :: ?(coffee, cup)`

Spreading activation from `coffee` and `cup` finds a relation they can both participate in, and this relation receives the most activation (ie `in`). Now, consider a more complex case, involving adaptation :

`on(container, plate) :: ?(stone, fire)`

Structural information leads to the candidate inference `on(stone, fire)`, but the retrieval mechanism causes substitution with the predicate `in(stone, fire)`. Two factors support this substitution, firstly `fire` and `stone` are already connected by an `in` relation (in memory), and secondly because the `in` and `on` share a common super-class (such as

spatial-relation). These two factors allow the source relation in to be substituted by the on relation. AMBR briefly describes two constraints that must be satisfied by transferred predicates. Firstly, its argument's must satisfy any argument restrictions, and secondly, the arguments should be found in the target domain. It would appear that finding appropriate arguments for a transferred predicate is derived from the spreading activation process. However, no details are provided on the constraints that apply to transferred predicates.

The evaluation phase determines the relevance and acceptability of any mapping, including partial mappings. Evaluation covers factors from consistency and validity to plausibility, relevance and applicability. Global evaluation is based upon the constraint satisfaction level. Invalid inferences for example, decrease the valuation placed upon the corresponding mapping.

The final phase of learning remembers the mapping by storing the relevant correspondence nodes, together with the excitatory connections. Finally, a generalised description of the mapping can be created by identifying the common super-classes of all the mapping elements.

2.8.1 Evaluation of AMBR

Clearly, AMBR is a very comprehensive model of analogy. It encompasses many phases that communicate through a shared memory. The retrieval process is semantically directed, using spreading activation to retrieve semantically local domains. However, the presence of appropriate goals and priming information may allow AMBR to retrieve semantically distant domains, and this opens the possibility of retrieving semantically distant and novel sources.

One general concern about AMBR stems from its very heavy reliance on spreading activation. Activation levels are simultaneously responsible for many operations: retrieval, initial mapping activation levels, and predicate adaptation. While other models use activation levels, no other model uses a single activation level to simultaneously support such a diversity of processes. The successful operation of AMBR may owe as much to the careful design of memory, as to the spreading activation process that operates

upon it. No sensitivity tests have been reported to explore this aspect of the AMBR model. This over-reliance on activation levels may easily be fooled by a memory that makes many indirect references to an irrelevant concept. Such a concept might always be active and force its way into all reasoning activities. For example, if much of memory is structured around say `food`, and the target references both `edible` and `inedible` products, memory will bias retrieval, mapping and adaptation - regardless of which source is identified. The absence of inhibitory links in memory to suppress this tendency, may compound this problem.

Transfer in AMBR relies heavily on background memory plus some concept hierarchy (presumably along with the source) to identify missing target information. AMBR never details the constraints that are applied to transferred predicates, so the exact mechanism that supported the rejection of the earlier `on(stone,fire)` predicate is not clear. For example, AMBR may have a problem representing `coal` as a `stone` that is `on fire` (ie `on(stone, fire)`). If the target relationship is a novel one, then it will not exist in memory and the mechanism for transferring novel predicates is also unclear. Generating novel inferences while rejecting invalid ones may prove challenging for AMBR and limits its ability to identify inferences that are both novel and useful. However, the notion of using argument restrictions to accept inferences is a very useful one, even though Kokinov provides few details on this process.

AMBR's adaptation and induction processes are primarily based on the contents of memory, combined with some concept hierarchy. However these processes are not always so dependent upon a pre-existing hierarchy, and adaptations do not always take the form of generalisation. AMBR makes much of its context dependent perspective on similarity. However, this is still constrained by its taxonomic restriction - similarity is only allowed between children of a super-class. So it does not generate new similarity, but merely identifies which form applies in the current context.

Because all phases in AMBR run in parallel and rely *so* heavily on spreading activation across memory, a deeper understanding of AMBR requires significant detail on the contents of its memory. Only then can we

truly study the principles that AMBR advocates, and separate these principles from the "accidental" influence that memory-contents have upon the models operation. AMBR is a very broad model of analogy encompassing the full spectrum of phases, but few specific details on its operation can be discerned to allow detailed examination and testing.

2.9 CopyCat

Copycat (Hofstadter, 1995) is a model of analogical reasoning for the micro-world of alphabetic character sequences. Architecturally, it is somewhat similar to AMBR, being a parallel model of interacting micro-processes called *codelets*. Interestingly, Copycat operates stochastically and so does not necessarily produce identical output on successive runs. As an example of the kind of problem addressed by Copycat, consider the following:

```
abc : ddab :: wxy : ?
```

Such a comparison has a clear reliance on the `successor` and `predecessor` relations, and also on the `first` and `last` letters of the alphabet. A key concept in Copycat is that there is rarely one correct answer to such analogies. Copycat's stochastic operation allows alternate codelets to operate upon the same problem data, producing different interpretations on each run. For instance, we could blindly replace the right hand side with the `ddab` sequence, and CopyCat infrequently produces this unlikely but credible answer. But this misses all the rich information contained in the source domain. In practice, CopyCat transfers the larger collection of relations describing the `ddab` sequence with a significantly higher probability, generating the `zzxy` answer much more frequently.

Three main architectural features support Copycat's operation; the *slipnet*, the *code-rack*, and the *workspace*. The *slipnet* memory-model stores relationships implicit in the domain, such as the `successor` relations between `ab` and `bc` letter pairs. *Codelet*s are small agent-like processes that operate on the problem space, identifying slipnet relations between letters, and between identified relations (like `same-as`, or `opposite-to`). The *workspace* repository annotates the letter sequences with identified slipnet

relationships. The workspace also allows codelets to utilise the output of previous codelets.

When presented with the earlier problem, codelets spread activation from the letters to the relevant relations in the slipnet, (most probably) identifying two successor relations between `ab` and `bc`. Other codelets identify higher-level structures, such as the identicality between the two successive `successor` relations. The slipnet might also identify opposite comparisons, for example noting that `z` has no successor often causes a `successor:predecessor` mapping between domains. Codelets also add other codelets to the code-rack, and execution of codelets is governed by a stochastic scheduling operation. Thus, each run of Copycat can result in a different execution sequence, producing the different results. One run could identify the "replace the right hand side with `ddab`" interpretation, while another might take the "identify the `successor:predecessor` relationship" reading of the analogy.

Joint focii of Copycat are the notions of conceptual distance and conceptual slippage (Hofstadter and Mitchell, 1988). Conceptual slippage is a feature found in more complex analogies, and is akin to the adaptation of inferences. For instance, slippage from identical relations (`successor:successor`) to opposite relations (`successor:predecessor`), forces the conceptual distance between opposites to be globally reduced. This is enforced by reducing the conceptual distance between these global concepts in the slipnet. Therefore, future operations on this analogy will favour comparisons between opposite concepts.

The global interpretation of the source is mimicked by the global interpretation of the target. Missing target relations are suggested by the mapping and are highlighted in the slipnet. Because these inferences effectively originate in memory, CopyCat does not use a separate validation process. That is, each individual inference in CopyCat is identified from the slipnet by the inter-domain mapping.

2.9.1 Evaluation of Copycat

CopyCat's operation is deeply embedded in the pre-existing memory of relations between concepts. Specifically, it does not assert *new* relations between two target objects that was not known before the analogy was presented. It focuses on identifying the combination of source relations that map to known target relations. All inferences are essentially triggered from memory, so Copycat does not generate inferences with the usual "pattern completion with substitution and generation" – this is referred to algorithmically as CWSG - Copy With Substitution and Generation (Holyoak *et al*, 1994). As Hofstadter (1995) says, *"Copycat does not model learning in the usual sense"*. CopyCat is focused on the plausibility of the mappings and its inferences, but is not concerned with the validity of inferences. Within the CopyCat memory, if an inference is plausible then it is considered valid. For example, the inference `h is-successor-of q` is unlikely to be considered (and will not be accepted) as it is not already contained in memory. Even if such a predicate were generated, this relation would not participate in other pre-existing relations to other concepts, and thus would not be maintained as a viable concept.

Although CopyCat does model long-term memory, it is domain specific and is not easily adaptable to general-purpose knowledge – nor adaptable to dealing with the ambiguity this entails. So CopyCat is primarily a model of the mapping phase of analogy and of aspects of domain access, but it is heavily bound to the micro-world of letter sequences.

2.9.2 Analogy Models using Only Target Objects

Another related segment of the analogy literature uses target domains that consist only of a list of objects - no target relations are specified. As well as the CopyCat model (Mitchell and Hofstadter, 1988; Hofstadter, 1995), other models operating on similarly represented problems include; *Analogy* (Evans, 1967), *TableTop* (French and Hofstadter, 1991), *Ludi* (Bohan and O'Donoghue, 2000) and *Cartographic Structure Matching* (O'Donoghue *et al*, 2003; O'Donoghue and Winstanley, 2001). These all solve analogy problems from "micro-domains" where all target problems consist only of

objects (see Figure 2.17). In these models, all source and target relations must be inferred from the two given domain descriptions. Interpreting the meaning of these domains plays an important role in solving these problems.

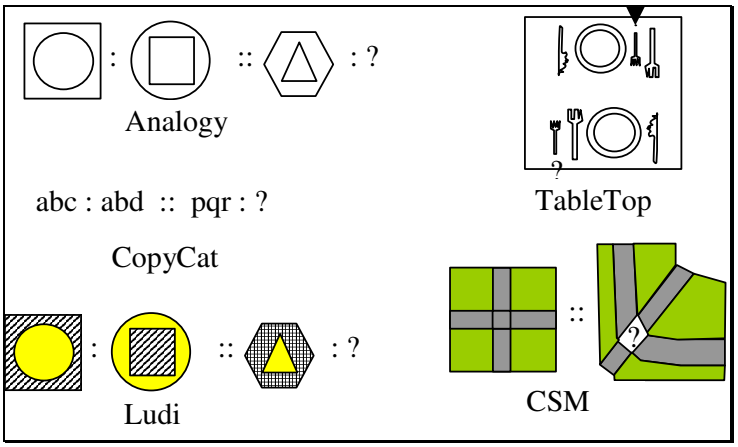

Figure 2. 17 - *Various target domains consisting only of objects*

We will briefly look at two of these models, as they have had an indirect impact on the current project. We begin with *Cartographic Structure Matching*[3] - CSM (O'Donoghue and Winstanley, 2001; O'Donoghue *et al* 2003). CSM categorises topographic map objects (ie polygons) into one of approximately 13 categories of object, based on simple outline maps of a region. The categories used include `building`, `road`, `rail` and `river` etc (though most data comes from just 6 of these categories).

CSM defines a number of templates against which problem data is matched, each template containing less than 10 adjacent polygons (see Figure 2.17). The (analogical) matching is done by the CSM, ensuring that the topology of the template and problem structures are identical. CSM also ensures that only categorised objects are mapped. When presented with a problem structure, it is the topology of the collection of objects that supports retrieval and mapping to the correct template (because so many templates

[3] This work was carried out in conjunction with the UK Ordnance Survey Research Centre, Southampton, UK. The work was partly funded by them and by the British Council.

contain the same polygons but in different configurations). In Chapter 4 we will see how topology is used to support retrieval.

The *Ludi* (Bohan and O'Donoghue, 2000) model is similar to Evans (1967) *Analogy* model. Ludi focuses on mapping and inference in geometric analogy problems involving polygons with attribute information, as each object can include several attributes like colour and pattern information (see Figure 2.17). Ludi identifies that attributes can play an important role in inference generation, and we will also return to this issue in Chapter 4.

2.10 Holographic Reduced Representations - HRR

All the models we have looked at so for are either based on symbol processing, are neurally inspired, or are based on the localist-connectionist framework. However, the Holographic Reduced Representations (HRR) (Plate, 1994; 1998) model of retrieval and mapping uses a completely different modeling technique. (HRR's were originally developed as a model of memory, representing compositional structures using a distributed representation - see Plate, 1991). HRR's incorporate semantic and structure together through a vector representation. Objects, predicates and predicate-roles are all represented by n-dimensional vectors, normalised to the unit hyper-sphere. The contents of an HRR vector encode four different types of information; *base vectors* (representing attributes of objects), *identity vectors* (representing a unique identifier for each object), *token vectors* (combined attribute and identity description of objects), and *role vectors* (predicate role information combined with unique object identifier information). A typical domain might be represented by (1, l, 0, 0, 1, 1, 1) /√5, where each element in the fixed-size vector represents information from one of the aforementioned categories. (The √5 normalises each vector description so that for example, large domains do not dominate during retrieval). The first three vector types represent semantics, while the fourth vector describes structure by the

combination of role-filler bindings. A process called *circular convolution*[4] approximates the structural-semantic similarity between two vectors.

Structure is maintained through weighted sums of convolution products. Binding-chains represent the fact that an object used in one role of a predicate, is also used in another role of a different predicate within the same domain. Binding-chains are further strengthened if the entities share similar attributes, *ie*. `Jane` and `John`, both of type `person`. During mapping, the presence of parallel binding-chains in both domains significantly strengthens the resultant structural similarity score. Semantically dissimilar mappings are generally weaker than semantically similar ones.

We consider the contextualised version of HRR's because of its superior performance. Consider the sentence *"Spot bit Jane causing Jane to flee from Spot*". HRRs convolve the various roles of the object with the object itself. So `Spot` is now contextualised as the `biter`, and as an object to `flee-from`. `Jane` is contextualised as something that gets bitten and as something that does fleeing.

HRR's require at least some semantic overlap between the target and the retrieved sources. However, HRR's do involve structure directly in the retrieval process (rather than in the later mapping phase like MAC/FAC and ARCS). So, domains with greater structural similarity are given higher retrieval scores than structurally dissimilar ones. However this can be counter-balanced by semantically similar domains bearing no structural similarity. So, HRR's perform adequately on retrieving within-domains analogies, but do not support retrieving between-domains analogies.

Because of the stochastic identification systems used by HRR's to identify entities within a domain, the same similarity score will not always be generated. So high retrieval scores are not guaranteed to accurately reflect structural similarity. Hence, retrieval scores are typically averaged over 100 simulation runs (Plate, 1998), though the standard deviation between runs

---

[4] Circular convolution is a variant of vector multiplication. Given two 1*n input vectors, it produces a 1*n sized output vector, instead of the n*n vector produced by standard vector multiplication. The "reduced" size of the output vector causes

appears quite low. Plate reports the following average similarity scores for the target probe P1: *"Spot bit Jane causing Jane to flee from Spot"*. In the source S1-S4 (see Figure 2.18) John and Jane are represented as `people`, `Spot`, `Rover` and `Fido` are dogs, and `Felix` is cat.

T1: *"Spot bit Jane causing Jane to flee from Spot".*	HRR	MAC
S1: Fido bit John causing John to flee from Fido.	0.71	1.0
S2: John fled from Fido causing Fido to bite John.	0.47	1.0
S3: Fred bit Rover causing Rover to flee from Fed.	0.47	1.0
S4: Mort bit Felix causing Felix to bite Mort.	0.30	0.6

Figure 2. 18 – *A Comparison of HRR retrieval scores*

Figure 2.18 shows the structural and semantic sensitivity of HRR's matching scores. It illustrates that the corresponding MAC retrieval score for S1, S2 and S3 is 1, and S4 is 0.6. The difference in retrieval strength for S1 and S4 illustrates that HRR's attempt to realistically model human analogising, combining structure and semantics in a combined similarity score.

2.10.1 Evaluation of HRR's

Plate reports that the maximum similarity score between two propositions that are both semantically and structurally identical is 0.80 (Plate, 1994). This seems surprisingly low, given that only the identities of the agent and patient objects differ between domains. Thus, the semantics expected of a useful source domain, make HRR's unsuitable for retrieving semantically richer sources from a very large memory.

Combining semantics and structure means that it is difficult to tell if a good mapping is the result of similar semantics and different structure – or vice versa. While in many uses this distinction is not particularly relevant, within creativity it is significant. Creative analogies often arise from semantically distant comparisons, but HRR give these analogies very low scores. If these creative analogies also supply inferences to the target

---

information loss and generally causes problems in computing the inverse of the

problem, then the similarity score will be reduced even further. This point is illustrated in Figure 2.18, where two very different source domains (S2 and S3) produce the same similarity score. In particular we note that all sources (S1 to S4) are structurally identical (isomorphic) to the target, and indeed each domain references one person and one domestic animal. From a creativity perspective, S4 is more likely to be overlooked by a person, and thus might even be favoured by a computational model as having greater creative potential than the more obvious sources. Thus, HRR's are not suitable for retrieving semantically distant sources.

Another limitation of HRR's from the creative analogising perspective, is its inability to retrieve sources with no semantic connection to the target. So while HRR's represent a significant improvement in analogy retrieval, their sensitivity to semantics limits their applicability to "far domains" retrieval. Limiting computation to vectors of predetermined size seems like a debilitating constraint but as AM, Eurisko and others (see Lenat 1983; Buchanan, 2001) demonstrate, much creativity can occur within a "micro-world". HRRs focus on mapping individual propositions, and do not directly map larger sets of domain descriptions. This serious limitation also affects the Drama (Eliasmith and Thagard, 2001) model that also makes use of Holographic Reduced Representations.

2.10.2 Drama

Distributed Representation Analogy MApper - Drama (Eliasmith and Thagard, 2001) is a mapping model that builds upon two pre-existing models; Holographic Reduced Representations (Plate, 1994) and ACME (Holyoak and Thagard, 1989). Rather than revisit both ACME and HRR's, we will just briefly describe some of the salient points of Drama.

In essence Drama is a variant of ACME that uses HRR's to generate the initial constraint network. HRR's compute the dot-product similarity between each source and the target proposition, keeping all results that lie above some minimal similarity threshold. Decoding the convolution vector identifies the *source:target* labels for the network nodes, while the similarity measurements

---

convolution function.

seed the activation value for each mapping node. Inserting inhibitory links and updating the activation levels are similar to the original ACME model. Drama provides no additional insights into the analogy process which are significantly different from either of the underlying models. As these have already been described in detail, we shall not describe Drama any further.

2.11 Overall Evaluation of Analogy Models

Constructing a model that can discover novel analogies, presents a new challenge to the modelling of the analogy process. Before we look at some of the requirements of the creativity model, we will briefly review these models to identify common approaches, and especially common shortcomings among these models. This review will focus on the requirements stated in chapter one, primarily related to the retrieval and validation phases.

2.11.1 Review of Mapping Models

Computational modelling of the analogy process began with the mapping phase, and the three original models: O/SME, ACME and IAM. The computational expense of O/SME and ACME led to the more tractable and cognitively plausible IAM (the later I/SME changed to IAM's incremental approach). The key to IAM's efficiency lies in its top-down search strategy that makes use of the pre-existing topology within the problem descriptions.

2.11.2 Review of Mapping and Retrieval Models

We examined five models of analogical retrieval and mapping; MAC/FAC, ARCS, Lisa, Sapper and HRR's. The first two are evolutions of earlier mapping models, adding a precursor retrieval process. Lisa, Sapper and HRR's combine these two phases in a more cohesive model.

Sapper deals only with access to the identified domains, and doesn't address the "spontaneous" retrieval (see Section 1.3.4) of candidate sources. However, its access strategy isn't bound by domain boundaries, allowing it to access semantically distant portions of memory. The other retrieval models perform semantically based retrieval, identifying sources that are similar to the presented target. MAC/FAC is the most constrained model, using it's content vectors to identify sources using the same predicates as the given

target. ARCS increases the range of identifiable sources by also identifying synonymous terms using the WordNet lexical database. Lisa can retrieve semantically distant sources, provided they reference the same semantic (attribute) units or sub-proposition units as the target. Like Sapper, Lisa is capable of retrieving quite dissimilar domains, although much of memory may be activated if overly general attributes are used (like "object"). HRR's are also semantically constrained, but again the semantic units allow quite distant domains to be reached. However, creativity thrives on semantically diverse sources (Boden, 1992), and none of these models explicitly seek semantically diverse sources.

Significantly, neither MAC/FAC nor ARCS make any use of topological structure during the retrieval phase (as opposed to during the mapping phase). Lisa considers structure in the form of predicate-role information, but Lisa's small phase-set size limits the amount of structure that can be considered at any one time. HRR's do consider structure during retrieval, although this is easily dominated by semantic similarity. Furthermore, it identifies structural similarity only when there is also some minimal semantic similarity between the two domains.

Several of the mapping models rely on active *mapping entities* to identify the inter-domain correspondence. The active mapping entities can be divided into two distinct categories. First, a common theme exists between the mapping models of Sapper, Lisa, and Astra. Each uses *mapping connections* that are asserted between potential mapping pairs, prior to the presentation of the mapping problem. These models use different techniques to identify which connections are implied by the given comparison.

Secondly, ACME and AMBR make use of *mapping nodes* (as opposed to mapping *connections*). These nodes too are created before the mapping is identified. Such an approach to mapping can generate vast numbers of mapping entities in a large memory, and so may not be entirely appropriate for our needs. Interestingly, many of these models do not address the topic of analogical transfer. This may be because no target items exist for the additional source material, and thus cannot participate in the mapping. This apparent difficulty with inference generation is a further indication that the

"mapping entity" approach may be inappropriate to our needs.

2.11.3 Review of Mapping, Retrieval and Validation Models

Finally, we compare models that encompass the phases of retrieval, mapping and validation, namely; CopyCat, Phineas, Astra and AMBR. Phineas' use of behavioural abstractions in retrieval allows semantically distant domains to be retrieved. But appropriate abstraction schemata are not generally available within a creative context, and so Phineas can not retrieve creative source domains. Phineas has an impressive model of verification that examines the consistency of inferences against previously known facts. This effectively constrains inferences at the inter-predicate level, but this comes at the cost of being highly domain specific. Therefore, extending this verification model to arbitrary target domains would require a comprehensive model of each possible target domain. Thus, Phineas' does not perform domain independent validation.

Copycat is designed to work only with a given source and target, and does not identify novel candidate source domains. Its "micro-world" approach to mapping and inference means that it does not do inference generation, in the traditional understanding of this term. AMBR and Astra both adopt a very pragmatic approach to inference, transferring inferences that are related to the current goal. But identifiable goals are not available within the creative context. AMBR does validate individual inferences, but the mechanism for this appears to be heavily influenced by previously recorded predicates. Thus, none of these models validate novel inferences for an arbitrary target domain.

2.12 Requirements of The Creativity Model

We now describe some of the issues that the proposed model will address, all of which differentiate it from the previously described models. As discussed in the last chapter, our focus is to create a model capable of discovering novel analogies. Such a model must necessarily encompass the phases of retrieval, mapping and validation. Our model (called *Kilaza*) requires a new approach in a few main areas. First it should identify semantically distant domains, as many of the reported creative analogies arise in such comparisons (Boden,

1992) and because distant comparisons may be more easily overlooked by a human analogizer. This will also allow Kilaza to explore different comparisons to a human presented with the same problem. Secondly, it should retrieve sources with the structural potential to supply inferences to the target problem. Third, we should allow mappings between semantically different domains, to maximise the number of analogies considered. Finally, it should reject any implausible inferences that are inadvertently generated. Validation should leave us with a small number of promising analogies and their inferences.

We expect that only a small proportion of the available sources will be capable of generating an analogy with the target problem, and only some of these will support useful inferences. We now briefly describe the novel requirements of our model, by examining each phase in turn.

2.12.1 Retrieval Requirements

In Section 1.3.4 we made a distinction between *access* to and *retrieval* of source domains, which we shall now refine even further. *Access* typically occurs under instruction from educators. The target and source domains are supplied, and may be described so as to highlight the analogical similarity. We also identify *spontaneous access* to a known source domain of target problem. Sanders and Richard (1997) discuss how people using a text editor, spontaneously retrieve the typewriter and handwriting domains. Finally there is *spontaneous retrieval* of a *novel* candidate source domain. Spontaneous retrieval of novel sources can be seen as the crucial driving force behind a model of creative analogising, as finding novel mappings and inferences is totally reliant on retrieval. We shall focus on the *spontaneous retrieval* of *novel* source domains, and henceforth use of the term *retrieval* will indicate such (unless otherwise indicated). Interestingly, Ritchie (2001) identifies *novelty* as one of the essential attributes of creativity, while novelty is also a necessary attribute of both *p-creative* and *h-creative* ideas (Boden, 1992; 1998).

We can identify two primary requirements of the retrieval phase of the Kilaza model. First it must be able to retrieve semantically distant domains

when presented with the target - because creative analogies generally arise from such comparisons (Boden, 1992). The focus on semantic features in retrieval seems to mimic people's performance on retrieval tasks in the laboratory, but is at odds with Dunbar's findings in more naturalistic settings. Blanchette and Dunbar (2000) and Dunbar (2001) report that people "very frequently" access deep structural similarities when presented with problems in naturalistic settings. Dunbar and Blanchette (2001) report that people regularly retrieve semantically dissimilar sources when generating analogies and formulating hypotheses. Therefore unlike existing models, we will not focus directly on semantics to support retrieval. The only semantics used will involve examining the target description for first-order relations, high-order relations and objects.

Secondly, we want to identify sources that are structurally capable of supplying inferences to the target. That is, there must be more material in the source domain than in the target. For example, if the target has four objects we will probably not be very interested in source domains involving only one object. Similarly, if the target predicates make many references to a single object, we should favour sources that are broadly similar in structure. Retrieval should also favour sources with more causal relations than the target. Of course, these must be soft constraints as we cannot guarantee that memory will hold any such domain, or that the target is free of irrelevant information. These requirements mean that existing models provide few lessons for our retrieval model.

Our model must also operate on a very large structured memory, with a potentially vast number of candidate sources. Thus, efficiency is still a major requirement of retrieval. The selection process must be discriminating enough to only identify a relatively small portion of memory, so as not to overwhelm the more expensive mapping process.

2.12.2 Mapping Requirements

The semantic diversity of the sources we must consider, means that mapping cannot be restricted by the predicate identicality constraint. Such a constraint would effectively reject the vast majority of candidate sources without

consideration. Thus, domain structure will be the only influence allowed on identifying the mapping.

As previously stated, our focus is on the inferences that mapping generates, rather than on the mappings themselves. Generating inference is normally achieved by the pattern completion algorithm known as CWSG - Copy With Substitution and Generation (Holyoak *et al*, 1994). In our model, each inference will undergo validation - rather than being accepted automatically.

2.12.3 Validation Requirements

Our validation model must be domain independent so that is can function for any given target problem. Because our creativity engine must work in what is essentially a learning context, we do not place pragmatic constraints on the inferences (Eskridge, 1994). Any (valid) inference in such a context will be accepted as we wish to acquire as much information as possible on the target problem.

Perkins and Salomon (1992) identify two different types of learning, called *near transfer* (or within-domains analogies) and *far transfer* (between-domains analogies). Kilaza will therefore be more interested in far-transfer than in near-transfer. The prime resource that is available to our validation process is based on the taxonomy that organises memory. We make use of the notion of *defined predicates* to support intra-predicate constraints. Defined predicates include additional information that specifies minimal attribute requirements for each role of that predicate. Only inferences that satisfy these attribute-constraints will be accepted as valid. So if a comparison mandates only invalid inferences, then the inter-domain mapping is rejected. So unlike Phineas which only uses inter-predicate based constraints, our model will be based on intra-predicate constraints derived from a taxonomy.

2.13 Conclusion

Modelling the analogy process has helped to focus research in the area, and allows competing theories to be compared. Many diverse techniques have been adopted to model the analogy process, though there has been a noticeable focus on the mapping stage. There is also a more recent trend

towards more comprehensive models of analogy, incorporating other phases with the mapping process.

Spontaneous retrieval has received some attention from the modelling community, though existing efforts have focused on the role of semantic similarity. This semantic similarity constraint is a limiting factor for a creativity engine, being blind to novel source domains that can generate novel inferences. Thus, creativity necessitates a retrieval model that overcomes this semantic bias, though preferably without suffering the penalty of "random" retrieval.

Analogical validation has received startlingly little attention from the computational modelling community. This can be at least partly attributed to the requirement of a large knowledge base against which to validate new information. There seems to be a need for some domain independent model of validation. However, the dependency between validation and background knowledge cannot be overlooked and may also impact the earlier phases of analogy. Creative analogising therefore presents some new challenges to the computational modelling of the analogy process.

Chapter 3

Kilaza's Memory

"It's a poor sort of memory that only works backwards."

- - Lewis Carroll, Through the Looking Glass, 1872.

"The real danger is not that computers will begin to think like men, but that men will begin to think like computers."

- - Sydney J. Harris, in "Return to Mathematical Circles", by H. Eves, PWS Publ. Boston 1987.

3.1 Introduction

Before we can understand the Kilaza model for identifying novel analogies, we must first examine it's memory structure. This memory is constructed around a taxonomic hierarchy that is linked to problem domain information. This taxonomy structures all relation, object and attribute information and allows the model to reason in depth about the problem structures.

Furthermore, the various phases of the model communicate directly via this memory which is used in a shared manner.

3.1.1 Structure of this Chapter

This chapter begins by examining the Kilaza taxonomy, which structures all problem domain and other concepts. We then describe how problem information is connected into the taxonomy. Then we examine how problem domains are stored and how Kilaza represents incomplete predicates, which are missing either a relation of some arguments. We introduce the notion of *functional attributes* and describe how they support role restrictions on relations. Finally, we take a brief look at the frame representation language used to represent the taxonomy and all problem information.

3.2 Memory Structure

At the most abstract level there are two components in memory, a *taxonomy* of concepts and a *predicate repository* containing recorded domain descriptions. The taxonomy specifies relations between the abstract concepts, effectively defining each in terms of its relation to the other concepts (Woods, 1975; Woods 1991). The predicate repository stores known domains, each being a thematically related collection of predicates - and includes all source and target domains.

The Unified Modelling Language diagram (Booch, Rumbaugh, and Jacobsen, 1999) in Figure 3.1 summarises the structure of memory. Firstly, the taxonomy is a collection of atoms arranged as an inheritance hierarchy, as shown on the left of Figure 3.1. Abstract concepts occur at the top of the hierarchy and propagate their values to the more specific concepts lower down the hierarchy.

Secondly, the predicate repository contains many domain descriptions. As shown on the right of Figure 3.1, each domain contains of a number of predicates. Each predicate then, is composed of a number of “instance nodes” representing the relation and arguments of each predicate. (Unlike Schank’s (1982) dynamic memory, we assume all information in contained with each domain description. Inferences are only generated by analogical comparison).

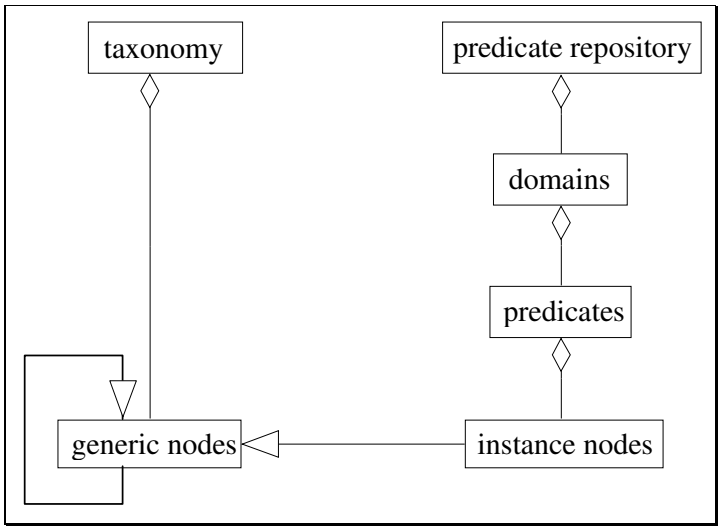

Figure 3. 1 - *UML diagram detailing the structure of memory*

3.2.1 Instance Nodes

For most of its operations, Kilaza uses only problem domain information. However specific operations refer to the taxonomy in order to clarify some part of the problem domain. Kilaza treats the taxonomy and the problem domain as distinct and separate entities, allowing the model to reason about each as independent (though inter-linked) entities.

The separation between the taxonomy and problem domains is achieved by using two different types of nodes. *Abstract* (generic) nodes are used only in the taxonomy and *instance* nodes are found only in the domain descriptions. *Abstract nodes* (eg `cat`) are either objects or relational predicates, and are defined by their position relative to the other nodes within the taxonomy. *Instance nodes* (eg `cat-171`) are found only in domain descriptions, and each instance node is connected to the corresponding abstract node. This distinction between *abstract* and *instance* nodes is central to Kilaza's operation, as we shall describe in the next chapter.

Consider the *solar-system* domain from Rutherford's analogy "the atom is like the solar-system" in Figure 3.2. (Note that Kilaza's `create-new-frame` function adds a domain to the predicate repository, and links each atom into the taxonomy). We generate a unique instance node for each atom in every domain, using lisp's `gensym` function. So, the predicate

`(attract sun earth)` is stored in long-term memory as something like `(attract1234 sun74375 earth43343)`. These instance nodes are linked to the generic concepts (`attract`, `sun` etc) in the taxonomy, thereby inheriting all properties of the generic concept. All occurrences of a node within a domain description are assumed to refer to the same entity, and thus use the same instance node. So, all occurrences of the `sun` object in Figure 3.2 will be given the same instance node number. Other domains using the same concepts will have different instance nodes (`attract678`). Kilaza supports a bi-directional linkage between all abstract and instance nodes. This allows our analogy model to move easily between generic and instance nodes, which simplifies many validation operations – as we shall see in Chapter 4.

```
(create-new-frame SOLAR-SYSTEM
     (PREDICATES
          (hot sun)
          (attract sun planet)
          (weight-difference planet sun)
          (and weight-difference attract)
          (cause and orbit)
          (orbit planet sun) ))
```

Figure 3. 2 – *Representation of the solar-system domain*

No restriction is placed on the concepts that can form instance nodes, so we can reason about instances of abstract objects (`living-entity74747`). Instance nodes are also used when Kilaza encounters unclassified objects, attributes and relations within a new domain description. These are assigned to the most abstract level in the relevant section of the taxonomy (*relation, object* or *attribute*) after examining the predicate's structure.

By the very nature of the process (and within Gentner's pure 'structure matching and transfer' perspective of analogy), analogy can never generate a completely novel relation as its inference, as every inferred relation originates in the source. If the source is familiar, then so are its relations making the

inferences possible to validate. Thus analogy is, almost implicitly, a self-regulating inference mechanism. If an inference is strong enough to be identified, it is capable of verifying its own validity (at least partly). This point will be explained in depth in the validation section of the next chapter.

3.2.2 The Taxonomy

The Taxonomy is used to structure and interpret the contents of all problem descriptions. Only the top levels of the taxonomy are used to support retrieval but the entire hierarchy is crucial to validation. The same memory structure represents background and working memory, but working memory is identified by having a positive activation level (though spreading activation is used by only a few processes).

The most abstract division in the taxonomy identifies the three main categories of `relation`, `object` and `attribute` (see Figure 3.3). Relations are further divided into `first-order` and `high-order` relations; `objects` are divided into the `physical` and `concept` sub-categories. (This taxonomy has been influenced by both Open-CYC and WordNet). Another sub-category of the `universal` concept is the `domain` class that stores all domains names, though these domain names do not form the core of the taxonomy. We point out that the taxonomy was developed as a general purpose taxonomy, and is a separate entity from the Kilaza analogy model. We will now look at the contents of the taxonomy in detail, as many of Kilaza's operations rely on it.

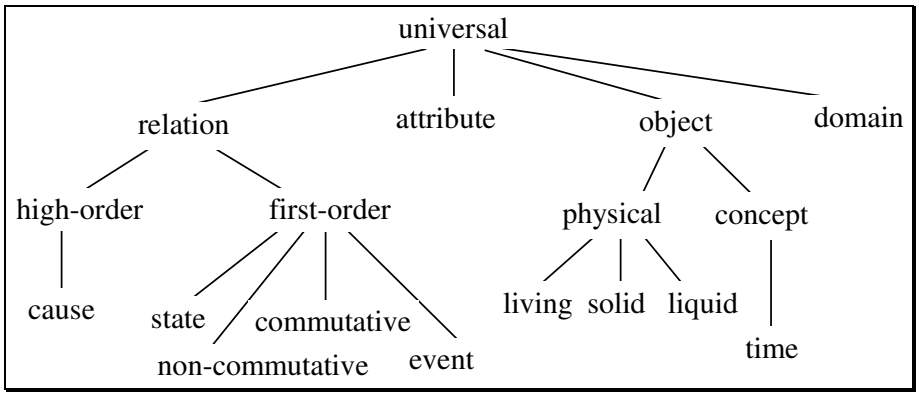

Figure 3. 3 - *The Top levels of the Taxonomy*

Relation Hierarchy

Relations are central to the structure of problem domains and we consider this segment of the taxonomy first. Relations represent an association between two or more arguments (almost all relations we consider are binary). As stated, all relations are categorised as either `high-order` or `first-order`. High-order relations represent relationships between other relations, and include `cause`, `result-in`, `inhibit`, `and`, `or` and `before`. Although causal relations are generally associated with spatio-temporal contiguity, Kilaza does not check for this.

First-order relations represent associations between objects. First-order relations are a primary concern of this model, as they are important to both retrieval and validation. Logical entailment is used to structure the hierarchy of first-order relations. The most abstract distinction separates `non-commutative` relations like `hit(a,b)` from `commutative` relations like `adjacent(x,y)` - this being equivalent to `adjacent(y,x)`. First-order relations are also categorised according to their temporal signature, as `events` or `states`. Events include `hit`, `drive` and `eat`, while states include `part-of`, `taller-than` and `president-of`. (Note: Kilaza does not explicitly identify transitive relations - `above(a,c)` *&* `above(b,c)` => `above(a,c)`).

Object Hierarchy

The second part of the taxonomy contains all objects and entities defined within the taxonomy. Objects are a sibling class to relations and are defined at the top-level of the taxonomy. Objects are divided into two types; the `physical` (`car`) and `conceptual` (`Tuesday`). Physical objects are further divided into `solid`, `liquid`, `gas` and `living-entity`. Physical solids are divided into the `mobile` (`devices` etc) and `immobile` (`buildings`) types. `Living-entities` are divided into `plants` and `animals`. Again super-classes represent more general concepts which are successively refined by subclasses, so the `automobile` object is successively refined by the following subclasses; `car`, `sports-car` and `super-car`.

Attribute Hierarchy

The final part of the taxonomy represents the attribute hierarchy. These attributes represent qualities that are used to describe objects. Attributes are divided into a variety of categories, including: `colour`, `temperature`, `shape`, `size` and `weight` (see Figure 3.4). Each attribute category is composed of an attribute value hierarchy, so `colour` is composed of `red`, `blue` etc; with `red` for example being further divided into `crimson`, `maroon` etc.

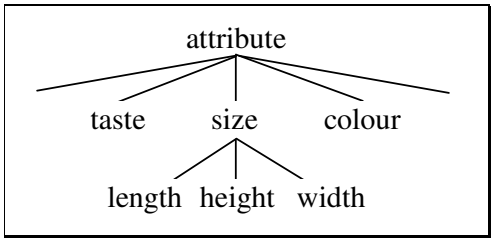

Figure 3. 4 - *A Segment of the Attribtue Hierarchy*

3.2.3 Using Attributes to Describe Objects

The taxonomy provides the framework that is used to describe objects and relations. Many objects have been predefined by attributes within the taxonomy. Objects are connected to their attributes using the `attr` slot that is associated with objects. Objects lower down the hierarchy inherit these attributes, while adding additional attributes. For example, all physical `objects` have the attributes `size`, `location` and `mass`. Some physical-objects have the attribute `mobile` while others are `immobile`. The living entity classes of `plants` and `animals` have the default attribute `alive`, and the `people` class has the additional attribute `intelligent`.

Objects can also dis-inherit attributes, allowing subclasses to differ from their super-classes. For example `birds` have the property of `flight`, but `penguins` and `emus` cannot fly; `mammals` live on `land` but `whales` do not and `people` are `alive` but `John-Doe` is not. Thus, atypical members are still represented as class members, but differ in the attributes they possess (Tversky, 1977). Dis-inheritance is supported by adapting the

inheritance mechanism using the additional slot `dis-inherit-attr`. When accessing the attributes of an object, Kilaza removes any dis-inherited attributes before the resulting values are returned.

Functional Attributes

Thus far we have seen the object, attribute and relation hierarchies and we have seen how attributes are connected to objects. The taxonomy also represents associations between the attributes of objects, and first-order relations (Figure 3.5). This connection directly supports the *functional attributes* that play a key role in one of the validation mechanisms we will see in Chapter 4. (These functional attributes are somewhat similar to the "functionally relevant attributes" mentioned by Keane (1985) and Eskridge (1994)).

Functional attributes specify necessary attribute requirements for each role of that predicate. Thus, functional attributes are intra-predicate constraints, ensuring that each predicate is a credible combination of a relation name and its arguments. For example, lets consider the relation `touch`, as in to "make physical contact with" (sense 1 of the verb `touch` from WordNet 1.7.1 online - See Appendix C). Both agent and patient roles of this relation must be physical objects, with the attributes `location`, `mass`, `height`, `width` etc. These functional attributes connect each role of a predicate directly to the attribute hierarchy, and arguments filling those roles must conform to these attribute constraints. Kilaza stores these functional attributes in the `agnt-attr` and `ptnt-attr` slots of the relation. Abstract relations typically have few functional attributes, but more specific relations accrue additional attribute restrictions.

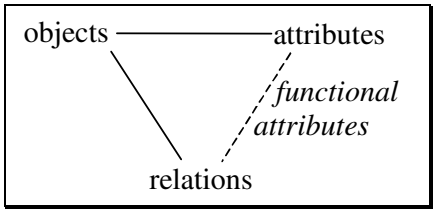

Figure 3. 5 - *Full Connectivity between Taxonomic Categories*

The relation touch is an abstract relation occurring towards the top of the relational hierarchy and it places relatively few restrictions on its argument types. Manipulate is a descendent of touch, and adds the additional constraint that its agent argument must also be alive.

Now consider the verb drive, as in to "control or operate a vehicle" (sense 1 of drive in WordNet 1.7.1 online - Appendix C). But drive inherits from the relation touch (via manipulate amongst others), which specify many of its argument restrictions. The agent role of drive adds the restriction of being a person, possessing the attributes alive and intelligent. The patient role adds that the patient argument must be mobile. Thus, the first of the following three predicates will be accepted, while the others will be rejected because they do not meet the functional attribute requirements.

```
drive ({alive, intelligent},
          {location, mass, mobile})
drive (bob, sports-car)
drive (mountain, sports-car) *
drive (bob, mountain) *
```

Consider the predicate kill(bob, cat) and (ignoring the polysemy of kill) assume that the agent role is restricted to animals via the attribute alive. The patient role of this predicate is restricted to any living entities, via the alive attribute. Now consider the predicate murder(bob, tom), whose relation inherits from the relation kill. Adding the functional attribute intelligent to the agent role supports the specialisation in the argument, and echoes the specialised relation itself. This framework underlies the functional attributes used for validation.

The entire memory structure used by Kilaza is stored in a frame based knowledge representation system. This will now be described, as shall the slots used to support the various memory structures described above.

3.3 Domain Representation in Kilaza

As stated earlier, Kilaza takes a domain description and internalises each node. Thus the contents of one domain always remain separate from other domains, even if they use the same tokens. However, all instances of a token (eg `man87` and `man93`) are linked to the one generic token contained in the taxonomy (see Figure 3.6).

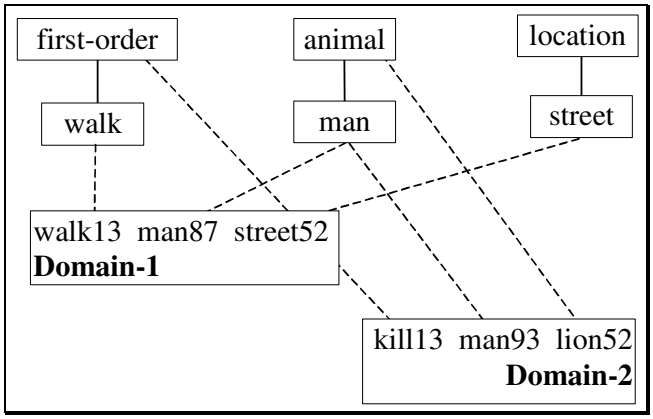

Figure 3. 6 - *All Instance Nodes Connect to the One Generic Node*

3.3.1 Partial Predicates

As described in Section 2.9, a distinct segment of the analogy literature uses target domains that consist only of objects - and includes no explicit relations (Evans, 1967; Mitchell and Hofstadter, 1988; French and Hofstadter, 1991; Bohan and O'Donoghue, 2000; O'Donoghue and Winstanley, 2000). In these models, the target domains consist of objects that act as cues.

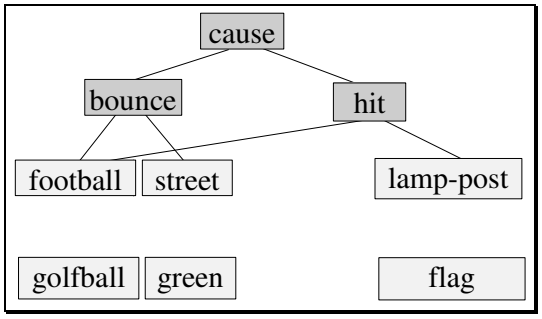

Figure 3. 7 - *Analogy between a Source and an Object-only Target*

From a conceptual view-point the source analog ("*The football bounced across the street and hit a lamppost*") can be viewed of as in Figure 3.7. This highlights the connectivity between predicates and objects and identifies the domain structure. However, problem domains may also be presented as partial (or incomplete) predicates. Thus, we wish to represent the following classes of under-specified predications:

- All target objects mentioned, but no predicates,
- All target objects mentioned, and some predicates,
- Some target objects and some target predicates mentioned,
- Some incomplete predicates in conjunction with some objects,
- Some target predicates mentioned, but no target objects.

These requirements mean that we must be able to represent incomplete predicates; such as objects not connected by a relation, or relations, that have no arguments. Although these may seem like trivial requirements, they do allow the use of a new range of analogs that have previously not been dealt with by one sector of the analogy literature. Thus we can talk about *target cues* as opposed to complete target predicates, enabling the creation of analogies when no target structure is present. Clearly, analogies without target structure must represent a challenge to the structure matching school of analogy, as we shall see in the next chapter.

```
(incomplete-target
     (predicates
          (cause)
          (bounce street)
          (football lamppost))
```

Figure 3. 8 - *A Partial Target domain*

In the domain of Figure 3.8 the `cause` predicate has no arguments, the `bounce` relation has only one argument, and the `football` and `lamppost` objects are not connected by any relation. When incomplete specifications are used in this manner, it is vital that the system knows which

atoms represent high-order predicates, first-order predicates, and objects. Without this knowledge the matching and inference processes would be too ambiguous. The same information is depicted in Figure 3.9, which indicates the partial structure that exists in the target domain.

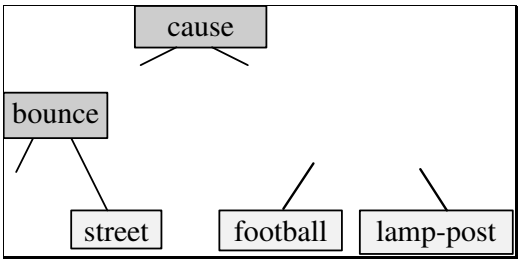

Figure 3. 9 - *Target information that forms incomplete predicates*

Many different target analogs can be mapped against the *"Football bounced across the street and hit the lamppost"* domain. It is up to Kilaza to infer any missing target information, using both domains and the combined information in the taxonomy and the predicate repository. The taxonomy provides the key to correctly interpreting these partial predicates, identifying the taxonomic category of all items in a partial predicate. This information allows Kilaza to search for the missing parts of the incomplete predicate. So, the partial predicate (`football lamp-post`) is identified as requiring a first-order relation to form a complete predicate. Furthermore, the functional attributes form a connection between objects and first-order predicates, enabling Kilaza to interpret many partial predicates appropriately. So, the partial predicate (`bounce street`) is identified as requiring an agent argument to form a complete predicate. Finally, the predicate repository can also be used to assist the process of interpreting the partial predicate and the analogy in which it originates. Thus, we may have greater confidence in a newly completed predicate if it resembles any predicates in the predicate repository. These aspects will be explored in detail when we examine the validation model (section 4.5).

3.4 The KRELL Language and the Background Taxonomy

Kilaza uses the Krell[1] Frame Representation Language to manage all problem information as well as the taxonomy. Kilaza uses a separate Krell frame to represent each domain, while slot-filler tuples represent the associations between concepts. Objects are connected to the attribute hierarchy with the `attr` slot and the attributes themselves use this slot to reference objects that use that attribute. Relations are connected to functional attributes with the `agnt-attr` and `ptnt-attr` slots. The `predicates` slot holds all predicates in the description of each problem domain.

3.4.1 List of Kilaza slots

The following is a list of the slots used by Kilaza, some of these slots are those used by the underlying Krell language while most are used by Kilaza. For example, the subsumption hierarchy uses the *children* slot to reference the subclasses, while the super-classes are stored in the *super* slot.

1. `Super` super-ordinate class - used by Krell for inheritance.
2. `Children` subordinate class - used by Krell for inheritance.
3. `Attr` slot of an object stores the attributes of that object. The `attr` slot of an attribute node lists the objects (instance nodes) that possess that attribute.
4. `Predicates` stores the domain descriptions as a collection of predicate calculus assertions. The target domain and all candidate sources are stored under the *domains* category at the root level of the taxonomy.
5. `Agnt-attr` functional attributes of the agent role of this relation.
6. `Ptnt-attr` functional attributes of the patient role of this relation.
7. `Activation` strength (a vector) for a node. If activation strength >0 then this node is part of working memory.
8. `Found-in` records every usage of each instance node, and the problem-domain that uses that node.

---

[1] KRELL (Knowledge Representation Entry Level Language) was written in 1993 by T. Veale and B. Smyth, Hitachi Dublin Laboratories, Trinity College, Dublin, Ireland, and was implemented in Common LISP.

3.4.2 Inheritance in Krell

Krell supports lazy inheritance between frames, and so slots and fillers associated with a frame are inherited by all subsuming frames. Upon accessing a slot, Krell performs a bottom-up search from the given frame and returns the filler of the first identified slot. More specific nodes only store extra information directly related to them, and do not duplicate information already specified at more abstract levels. So, if *mammals* are described *warm-blooded* and if *carnivores* are a type of *mammal* defined as *flesh-eatings*, we only store the *warm-blooded* information once as it is accessible from the *mammal* node via the hierarchy. Thus, when accessing slots we must combine the fillers of all parent slots to return the full result.

For example, all objects have a `size` and `location` attribute. Qualities associated with objects are also stored in the attribute hierarchy, and reverse links are maintained supporting retrieval of the corresponding object when supplied with the attribute and vice versa. As we shall see, this two-way indexing is particularly useful to Kilaza's validation operations.

Kilaza allows multiple inheritance and neither checks for nor enforces consistency between inherited values. While this can potentially result in conflicting information being inherited (Touretzky, 1986; Thomason and Touretzky, 1991; Sowa, 1992), this is not a problem in practice. Kilaza searches for the presence of attributes associated with an object (say), and ignores any contradictory information. However, the taxonomy has been designed so that such conflicts rarely occur.

Krell supports inheritance but not automatic classification, contradiction detection or other more advanced knowledge representation operations (Brackman *et al*, 1991). We point out that Krell does not directly support role restrictions as implemented by languages like KL-ONE (Brachman and Schmolze, 1985) and Classic (Brachman *et al*, 1991). Role restrictions are supported using *functional attributes* that are stored as normal Krell slots, although they are used in a very special manner in the knowledge base. So Krell's advanced features like dynamic data links (or *hot-links* in Krell terminology) are not used. Finally, memory does not use weighted association

between concepts. Thus, our basic memory model is quite simple, keeping the focus on the analogy model.

3.4.3 Some Relevant Krell Access Functions

Krell supports all of Kilaza's memory based operations, storing the taxonomy and all problem domains. Kilaza's basic Taxonomy is defined in the file *background.l*, which generates the class hierarchy. Reasoning within Kilaza relies mainly on the following Krell functions.

- `Krell-get-value` *`frame slot`*: Retrieve the *first* value contained in the *slot* of the *frame*. If no local value for this exists, then perform an inverted depth-first search up the taxonomy to find the required slot in a more general frame. This Krell function returns the first element if the result is a list.
- `Krell-get-values` *`frame slot`*: Retrieve the *entire* contents of the *slot* of the *frame*. If no local value for this exists, then use an inverted depth-first search up the taxonomy for the required slot.
- `Krell-get-local-value` *`frame slot`*: Like `Krell-get-value` but it does not resort to inheritance.
- `Krell-set-value` *`frame slot filler`*: Define the contents of the *slot* of the *frame*, with the value *filler*. Filler may be either an atom or a list (or any other Lisp) structure.
- `Krell-set-values` *`frame slot filer1 filler2 ...`*: Define the contents of the *slot* slot of the *frame* frame, but it does not check for duplicate information within the *filler* list.
- `Krell-add-value` *`frame slot filler`*: Add another value onto the contents of the *slot* of the *frame*. This may result in duplicate information if the new information already exists.
- `Krell-replace-value` *`frame slot old-filler new-filler`*: Replaces the value of the *slot* with a new value.

Many low-level routines in Kilaza make direct use of these Krell functions. The more high-level functions in Kilaza use Kilaza's own utility routines,

rather than using Krell directly. This means that the Kilaza model can be easily migrated to work on a different knowledge representation language.

3.5 Conclusion

Memory is central to any extended model of the analogy process. We described Kilaza's two-part model of memory, encompassing a taxonomy and a predicate repository. Problem domains are represented as frames of thematically related information and all information is linked to the corresponding abstract concepts in the taxonomy. The underlying taxonomy represents the relationship between abstract concepts and a dis-inheritance mechanism allows atypical class members to be represented.

Functional attributes are used to represent role restrictions on the arguments of defined relations. These specify minimal attribute requirements for each role of a defined relation. These are used primarily during the validation phase, which will be descried in the next chapter. Finally, we saw how Kilaza represents partial predicate information on a target domain - corresponding to target cues on a partly known target domain. We can now progress to examine the analogy model itself.

Chapter 4

A Model for Identifying Creative Analogies

"We should not introduce errors through sloppiness; we should do it carefully and systematically."

-- Edsger Dijkstra, in A Discipline of Programming, Prentice Hall, 1976.

"T'ain't what you do, it's the way that you do it"

- - Oliver and Young, 1936

4.1 Introduction

Computational models have made a significant and continued contribution to the study of analogy, as discussed in Chapter 2. Many of the models highlight specific influences on the process, and modelling allows researchers to ensure the correctness of their theories. Indeed, comparison between models forms an important part of the analogy literature (Keane, Ledgeway and Duff, 1994; Law, Forbus and Gentner, 1994; Veale, Smyth, O'Donoghue and Keane, 1996; Veale and Keane, 1997; Forbus, Gentner, Markman, Ferguson,

1999). In this chapter we take up the challenge of developing a multi-phase model of analogy, that is capable of discovering novel analogies.

As stated in Chapter One, semantically distant source domains play a central role in creative analogies. Retrieving these semantically distant sources is the first of three key challenges that will be addressed in this chapter. We present a retrieval model that attempts to find potentially useful sources, and which overcomes the usual semantic bias favouring sources that are similar to the target problem. We describe a mapping model that can generate mappings between these diverse sources and the target. However, this is just a modification of the "standard" incremental mapping model, and is not a particular focus of this thesis.

Creative analogies also present a challenge to the validation process. Validation must identify and reject any unwelcome inferences that these creative analogies generate. We present a validation model, which operates on the inferences that are generated by the creative analogical comparisons. The validation process must accept the plausible inferences while rejecting incongruous ones. We use these inferences to distinguish between invalid analogies (which are rejected) and those that might be potentially useful to some reasoning agent.

Hence, this chapter addresses three phases of analogy. First, we describe the model of analogical *retrieval* that focuses on a domain's graph-structure rather than a domain's semantic content. This increases the semantic diversity of the candidate sources that are identified. Secondly, we describe a variant of incremental mapping, which also makes use of a domain's graph-structure to simplify the mapping process. Finally, we describe a *validation* model that rejects many invalid inferences, and also rejects the driving analogies where appropriate.

One issue that runs throughout this chapter concerns the "predicate identicality[1]" constraint used by SME (Falkenhainer, Forbus and Gentner, 1989) and MAC/FAC (Forbus, Gentner and Law, 1992). Early testing on the

[1] Only identical predicates can be mapped between the two domains.

model revealed that this could not be employed as a hard constraint, because this frequently left us with no mapping and no inferences. (Re-representing the domains with a common vocabulary will be discussed in detail the end of Chapter 5). Abandoning identicality as a *hard* constraint improves the models ability to retrieve, map and validate analogies between semantically distant domains or between domains described using different terminologies.

4.1.1 Structure of this Chapter

This chapter contains three sections, and the first describes our model for analogical retrieval. This section also assesses the implications that the presented model has for both mapping and validation. Secondly, we briefly examine our model of analogical mapping, describing it as a variant of *incremental* mapping (cf Keane, 1990). Thirdly, we describe the validation mechanisms used in Kilaza and its accompanying adaptation model.

4.2 Kilaza Overview

Before we examine the individual phase models, we present an overview of the Kilaza model. It is a three-phase model encompassing *retrieval, mapping* and *validation*, although its main foci are on retrieval and validation. Both retrieval and validation models rely heavily on the model of memory that was described in the last chapter. The architecture of the Kilaza model is depicted in Figure 4.1, showing the models of each of the three phases as well as the underlying memory model. The remainder of this Chapter describes each of the phase models in turn, describing their interaction with each other and with memory.

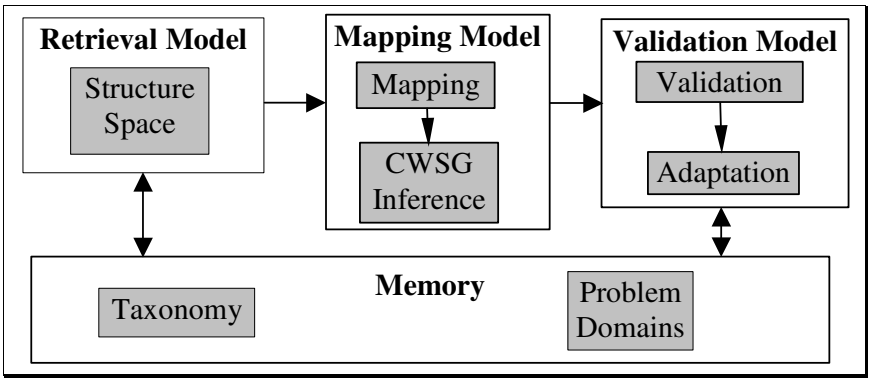

Figure 4. 1 - *Architectural Overview of the Kilaza Model*

4.3 Structure- Based Retrieval

Novel analogies thrive on having a diverse supply of source domains that could be used to interpret the presented target domain. This need for diversity places a unique requirement on the retrieval phase that is often ignored in current models; Kilaza needs to explore sources that are semantically distant from the target. However, such a model should not retrieve too many domains, especially domains that are incapable of participating in a mapping to the given target. We present the Kilaza model designed to satisfy these requirements, making it quite different to other models.

We identify two distinct sources of similarity on analogue retrieval, which we illustrate using the *solar-system:atom* analogy (see Figure 4.2). First, the *semantic similarity* can be characterised as the search for domains containing the same predicates as those used in the target. This is usually referred to as the *predicate identicality* constraint adopted by MAC/FAC (Forbus, Gentner and Law, 1995). So, if the target contains the relations `attracts` or `heavier-than`, then we search for other domains using these tokens. Predicate identicality can be seen as too restrictive and ARCS (Thagard, Holyoak, Nelson and Gochfeld, 1990) relaxes this constraint to also identify similar (rather than just identical) terms. This is achieved using a taxonomy to identify synonymous terms, like `more-dense-than`, as well as more abstract terms, like `weight-difference` or `tangible-object-relationship`. (Later in this chapter we discuss the inherent limitations with this analogy - see section 4.5).

Figure 4. 2 - *Semantic Similarity of the Solar-system and Atom Domains*

Secondly, we identify the *structural similarity* between domains. This is characterised as the search for domains that are either isomorphic or homomorphic with the given target description. In this thesis we will use the term *structure* to indicate the graph-structure of a domain's description (see Figure 4.3). The objects and relations form the nodes of this graph structure while the predicate and argument structure forms the edges of this graph. Nodes are classified as either objects and relations, while relations are further categorised as first-order or high-order relations to help identify structurally similar domains. However, the details of this structure perspective will become clear later in this section. So, the structure of the *atom* domain may be summarised as a domain having two objects and two non-commutative relations. An important part of the graph-structure of a domain concerns its argument structure. In the *atom* domain, one object is the agent of both relations and the other object is the patient of both relations (see Figure 4.3).

We therefore wish to identify source domains that are structurally similar to this target, as only these domains can form a useful mapping with the target. HRR's (Plate, 1998) not only uses semantic similarity in its searches for similar predicates, but also searches for predicates with a similar argument structure to a given target. However, HRR's cannot identify similar predicate structures in the absence of any identified semantic similarity. Interestingly, ARCS (Thagard *et al*, 1990) uses synonyms *etc* to adapt the domain semantics, but the domains graph-structure remains unchanged.

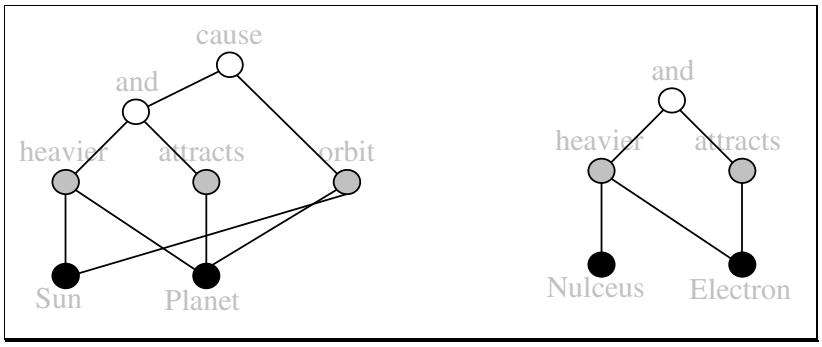

Figure 4. 3 - *Structural Similarity between the Solar-system and Atom domains*

A Two-Part Retrieval Model

We present a two-part model of analogy retrieval (see Figure 4.4) consisting of independent *structural* and *semantic* components. Structure-based retrieval focuses on the graph-structure of a pair of domains. We can be support standard analogical retrieval by combining these metrics, so that identification of source domains takes both factors into account.

$$Analogical\text{-}similarity \approx f(semantic\text{-}similarity,\ structural\text{-}similarity)$$

This states that the analogical similarity between a pair of domains can be estimated by a function (*f*) of their semantic and structural similarity. However our creative requirement means that the semantic similarity component has not been implemented in this model, as it will tend to reduce (rather than increase) the diversity of identified sources. So the metric we use to support the identification of novel analogical comparisons is:

$$Novel\text{-}analogy\text{-}similarity \approx f(structural\text{-}similarity)$$

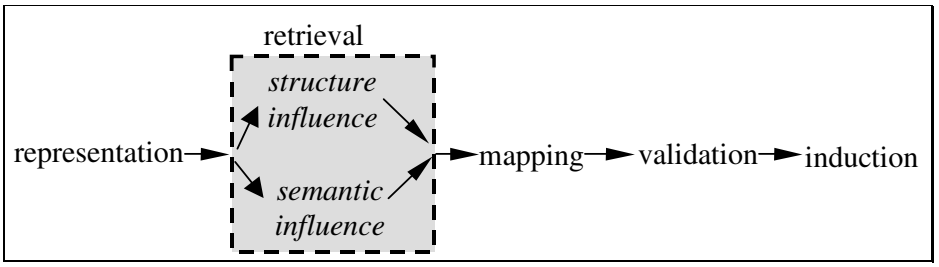

Figure 4. 4 - *A Two-Part Model of Analogy Retrieval*

The structural similarity process uses the graph-structure of the target domain to retrieve structurally identical (isomorphic) and structurally similar (homomorphic) candidate sources. This treats the target domain as a graph and attempts to identify sources that have the same or similar structure to the target. The objects and relations of a domain become the nodes in the graph, while the combination of relations and their arguments yields the edges of that graph. Thus, the retrieval model used by Kilaza is based on the graph-structure of the presented target domain. (The retrieval algorithm would require only minor modifications to cope with changing the representation of predicates from nodes to edges - ultimately producing the same results).

Structure in Retrieval

Mapping is the core of analogy, and a retrieval model must identify sources that can form a viable mapping with a given target. But analogical mapping is a variant of the Largest Common Subgraph (LCS) problem (Garey and Johnston, 1979; Veale, O'Donoghue and Keane, 1995; Veale, Smyth, O'Donoghue and Keane, 1996). LCS is the problem of finding the maximally sized graph-structure that is common to two presented graphs. The IAM mapping constraint (Keane, 1990) requires that over half of the target items participate in a mapping before that mapping is considered viable. So when searching for a candidate source that maps with all (or most) of the target, we must identify sources that are structurally similar to the target. If *all* of the source and target domain map completely together, then their structures must be isomorphic. If *most* of the two domains participate in the mapping, then their structures must be homomorphic. Therefore, analogy retrieval becomes the problem of identifying isomorphic (and homomorphic) structures to some

presented problem. (We will address the distinction between isomorphic and homomorphic retrieval later in this chapter).

The following example highlights the role of structure within a domain's description (O'Donoghue and Crean, 2002). Consider the domains described in Figures 4.5 (a) and 4.5 (b), and the problem of distinguishing between them for the purposes of analogical retrieval. From a semantic perspective, both domains contain three instances of the "loves" predicate plus three arguments representing people.

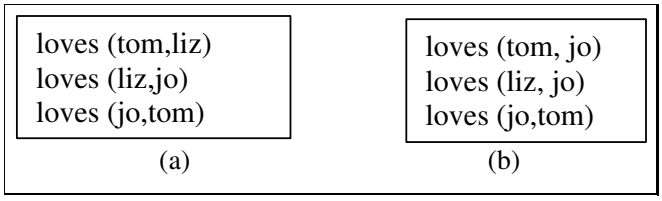

Figure 4. 5 - *Two semantically similar domains*

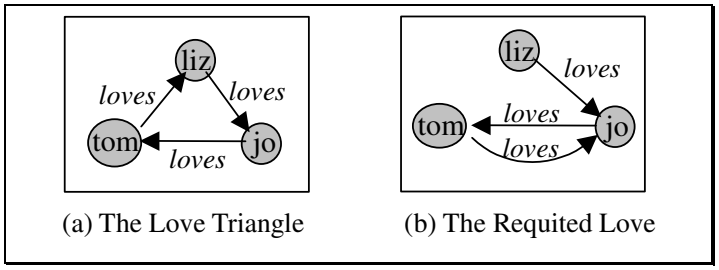

Figure 4. 6 - *Structural perspective on domains 4.5(a) and 4.5(b)*

Now consider these domain descriptions from the perspective of their (graph) structure. The implicit triangular structure of Figure 4.5(a) is illustrated in Figure 4.6(a) - the "Love Triangle" domain. The implicit non-triangular structure of Figure 4.5(b) is illustrated in Figure 4.6(b) - the "Requited Love" domain. The structure of these two domains is *central* to their semantics - even the name of the love-triangle domain reflects its structure. Altering the structure of either domain results in altering the meaning of that domain. So the meaning and structure of these domains are inter-dependent, a factor that structure-based retrieval aims to exploit. Indeed

for these examples, domain structure successfully distinguishes between the domains while the traditional "semantic similarity" perspective used by MAC/FAC (Forbus et al, 1995) and ARCS (Thagard et al, 1990) would not.

Consider also the two creative analogies used by Kekulé in Chapter 1. In his first comparison, Kekulé identified an analogy between a line of carbon atoms and the links of a chain. We argue that the structure of the two domains offers greater explanatory insight into understanding the origin of this analogy - focusing purely on the semantics of the problem appears to be a less promising avenue for credible investigation.

Kekulé's subsequent invention of the carbon-ring appears to involve two very different structures - a linear snake and a ring. However, this analogy arose when dreaming of the linear snake biting its own tail, so the source domain contained both the linear and ring structures. Kekulé merely applied the structural transformation of the source domain to the target domain of the carbon-chain, and this resulted in the carbon ring. On the basis of these arguments, it seems possible that structure might play a significant role in analogy retrieval - and in finding novel and creative analogies.

4.3.1 Features of Structure

More specifically, we use "features of structure" to characterise the structure of all problem domains. These are simple numerically-based features derived directly from the *representation* of each domain (rather than being derived from the problem domain itself). Structural features treat each domain representation as a graph, and describe elementary structural features of that graph (Figure 4.3). For example, we count the number of predicates found in each domain, and thereby favour similarly proportioned candidate sources. The (abstract) taxonomic classification of atoms is necessary to calculate some of these structural features. This requires access to the taxonomy to distinguish between, for example, first-order and high-order relations.

The structural features actually used by Kilaza are:

1) *Number of object references* – This counts the number of times objects are referenced within the domain description. Retrieval may distinguish between domains containing just a few object references from those that rely on large collections of objects.

2) *Number of unique objects* – This removes duplicate references to an object, to identify the number of unique objects used within a domain.

3) *Number of first-order predicates* – First-order predicates are predicates that take objects as arguments. They often form the bulk of the information that participates in a mapping, and thus is of central importance to analogical retrieval. Retrieval will search for candidate sources with similar numbers of first-order relations.

4) *Number of unique first-order predicates* - This differs from the "number of predicates" by removing duplicate predicates from consideration. This distinguishes between domains that rely on repeated use of a few predicates, from domains that use many different predicates.

5) *Number of root predicates* – A root predicate in a domain is a predicate that is not an argument to another predicate, and they typically represent the controlling causal structure within a domain. Root predicates play a crucial role in incremental mapping models like IAM and I/SME. Counting the number of root-predicates will facilitate identification of similarly structured sources.

6) *Maximum object usage* - This counts the frequency that each object is used by non-commutative relations. This measure explores the intuitions that some objects will be heavily referenced whereas others may only be referenced once. Kilaza does not count the arguments of commutative relations when calculating the *Maximum object usage* values, because including one representation would dis-favour the retrieval of the commutative form. However, these relations are counted for the *Number of first-order predicates* and the other features of structure.

7) *Number of high-order predicates* – This feature of structure describes the amount of causal structure that is contained in each domain. Useful high-

order relations include `cause`, `result-in`, `leads-to`, `and`, `xor` and `but-not`.

Homomorphic retrieval is the prime goal for analogy retrieval, as we expect the source domain to contain more causal information than the target. Figure 4.3 depicts the difference in structure between the *solar-system* source and the *atom* target, and the additional causal material of the source is clearly identifiable.

The features os structure listed above were designed to support both isomorphic and homomorphic retrieval. Of course this is not an exhaustive list of all possible structural features, but these features do serve to distinguish between varieties of domain structures. Additional features might include identifying loops, forks and joins within the relational structure, or the ratio of predicates to objects *etc.* More fine-grained classifications could count the number of *event* (*eg* `bounce`) and *state* (*eg* `taller-than`) relations in a domain, or describe the number of *physical* (*eg* `house`) and *conceptual* (eg `Tuesday`) objects in a domain. Each feature may be more or less useful depending on the domain descriptions involved (Crean and O'Donoghue, 2001, 2002; Crean, 2001, 2003). The remainder of this thesis will rely on the features of structure that are listed above.

4.3.2 Structure Space and Isomorphic Retrieval

Structure based retrieval is achieved by first mapping the representation of each domain into an N-dimensional *structure space* (O'Donoghue and Crean, 2002). The dimensions of this *structure space* are the 7 structural features described earlier - such as *Number of first-order predicates* etc. The value of each structural feature is calculated in turn for each domain, and the domain is located at the appropriate location in structure space (see Figure 4.7). First, we will describe how structure space is used to support isomorphic retrieval. Then we extend this technique to support homomorphic retrieval, and finally we describe its use in analogy retrieval.

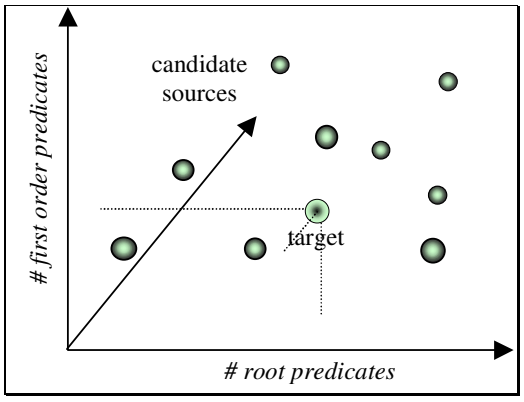

Figure 4. 7 – *Locating domains in structure space*

Isomorphic Retrieval

Any two isomorphic structures will necessarily have the same values on each of their structural features. When all candidate sources are mapped into structure space, isomorphic retrieval is achieved by identifying all domains at the same location as the target problem. Of course, only a complete set of structural features will guarantee the isomorphism of co-located domains, and the set of structural features listed above is not guaranteed to be complete. However, if isomorphic sources exist then they will be co-located with the target in structure space.

If we return briefly to the *love-triangle* and *requited-love* domains introduced in Figure 4.6, we can see that the Features of Structure distinguish between their structures (see the "Maximum Object Usage" row in Table 4.1).

Furthermore, if we presented the *atom* problem to a memory containing many candidate sources (including the *solar-system* domain), it would identify any structurally identical candidate sources. As stated earlier, analogy retrieval does not look for isomorphic domains, but rather looks for sources with additional causal structure. Thus, we will not discuss isomorphic retrieval any further.

	Love-Triangle domain	*Requited-love Domain*
Number of object references	6	6
Number of unique objects	3	3
Number of first-order predicates	3	3
Number of unique first-order predicates	1	1
Number of root predicates	3	3
Maximum object usage	2	3
Number of high-order predicates	0	0

Table 4. 1 – Structural Features for Two Domains

4.3.3 K-Nearest Neighbours and Homomorphic Retrieval

For any given target analogue, the desired source domain must contain the additional material, which will form the inferences to that target. This transferable material makes candidate sources homomorphic to (rather than isomorphic with) the given target domain. This transferable material will typically be in the form of some causal relations and perhaps some first-order relations. If structure space is to support analogy retrieval, it must therefore support the retrieval of homomorphic candidate sources.

Kilaza uses the Nearest Neighbours algorithm within structure space to identify these homomorphic domains. The nearest-neighbours algorithm allows the identification of domains with similarly valued structural features, to the given target. This allows the identification of the desired candidate sources, with their additional causal and other relations. However, use of the nearest neighbours formula also eliminates negative values, which can make two domains seem closer than if a metric incorporating negative values were

used. Additionally, some attributes might inter-correlate with a further impact on the similarity metric. Identifying which attributes most successfully support retrieval would require a more thorough investigation using an extensive database of analogies, but lies beyond the scope of this thesis. The Euclidean distance between domains in structure space is defined in Equation 4.1.

$$d = \sqrt{(ta - s_x a)^2 + (tb - s_x b)^2 + (tc - s_x c)^2 \ldots}$$

Equation 4. 1

where: d is the distance between the target and the candidate source domain, a,b,c are the various the structural features listed in Section 3.4.1, ta is the target's value for structural feature a, and similarly for tb, tc *etc*. S_x refers to the x^{th} source domain, so $S_x a$ is structural feature a for source domain x, and similarly for $S_x b$, $S_x c$, *etc*.

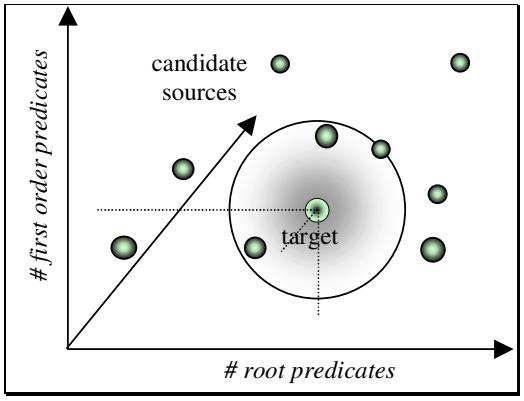

Figure 4. 8 - *Homomorphic Retrieval in Structure Space*

Firstly, all candidate sources are mapped into structure space, a process that identifies a 7-tuple of structural features with each candidate source. Next, the structural features of the target are identified. Finally, the Euclidean distance between the target and all candidate sources is calculated. (This requires examining each domain in turn, but Crean (2003) has explored the use of spreading activation to reduce to cost of this part of the algorithm).

Only those domains within a certain threshold distance are returned as the candidate sources from the analogy retrieval phase (see Figure 4.8).

Structural Features for Analogy Retrieval

We now present the final modification to structure space retrieval, which tailors it specifically to the problem of analogical retrieval (rather than supporting generic structure-based retrieval). Equation 4.1 favours domains whose structures are most similar to the given target. Thus, sources with extra causal structure are found at the same distance from the target, as sources with *less* causal structure! Clearly this is not an ideal way to support analogy retrieval.

To rectify this problem, the structural features of the target are directly manipulated from their original values. These manipulations account for the additional causal and other structures that we want to "carry over" to the target domain. For example in the *solar-system:atom* analogy, the additional source information consists of one causal relation (`cause`), one non-commutative first-order relation (`orbit`) and one commutative high-order relation (`and`). To favour the retrieval of these source domains (and to disfavour source domains with less causal structure), we displace the *locus-of-retrieval* with respect to the targets position (see Figure 4.9).

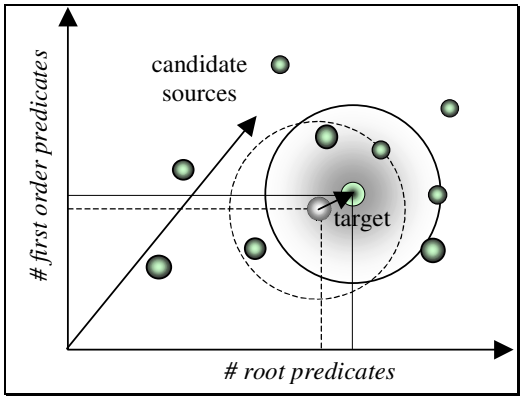

Figure 4. 9 – *Modifying the Locus of Retrieval to Identify More Appropriate Source Domains*

The following intuitions form the basis for modifying the locus of retrieval in structure space.

i) The source has more first-order relations

ii) These first-order inferences refer to more objects

iii) The source has more causal relations

iv) The source has fewer root predicates (because of the additional causal structure)

Thus, retrieval is biased in favour of candidate sources with the potential to supply candidate inferences. So, we modify the locus of retrieval by adding the expected structure of the inferences to the target's location in structure space. Therefore, the locus used for structure-based retrieval is as follows:

1) Number of object references + 1

2) Number of unique objects

3) Number of first-order predicates + 2

4) Number of unique first-order predicates

5) Number of root predicates – 1

6) Maximum object usage

7) Number of high-order predicates + 1

This is clearly an heuristic rule and can be easily modified. However initial testing on a select sample of well-known analogies showed that these modifications generally had a positive effect on reducing the inter-domain distance (within structure space) between source and target (see Table 4.2). (All domain descriptions are listed in Appendix A). However, these modifications did not universally reduce the inter-domain distance, as can also be seen from this table.

	Source to Target Distance within Structure Space	Source to Locus-of-Retrieval Distance within Structure Space
Atom:Solar-system	8.24	6.08
Heat-flow: Water-flow	6.92	4.79
Kennedy-Saga: Arthurian-Saga	5.38	4.47
Tumour:Fortress	0	2.64
Atom-falkenhainer: Solar-system-falkenhainer	3.87	3.464

Table 4. 2 - *Effects of Modifying the Locus-of-Retrieval*

Scaling the Axes in Structure Space

All structural features are defined as being of equal importance in Equation 4.1. A further modification that could be made is to scale the relative importance of each feature in our distance equation. We define P, Q, R… to be scaling factors in Equation 4.2, making structural feature *a* P-times as important as the other features. Thus, any difference between a candidate source and the target based on this feature value will have a greater influence on retrieval probability (if P>1). If a feature isn't significant for retrieval we make P<1, while P=0 removes this structural feature from consideration.

$$d = \sqrt{P(ta - s_x a)^2 + Q(tb - s_x b) + R...}$$

Equation 4. 2

Determining the best values for the scaling factors P, Q, R… would be best achieved if a very large corpus of known analogies were available to

determine the best values for P, Q, R. In the absence of a principled reason for modifying the values P,Q, R, the un-scaled distance formula must be used (Equation 4.1).

4.3.4 The Retrieval Algorithm

Retrieval is a two-part process; firstly populating structure space with the candidate sources and secondly retrieving sources from that space. Separating the population of structure space from retrieval means that structure space is created only once, and can support numerous retrieval episodes thereafter. Creating structure space involves iterating through each candidate source and calculating the value of each structural feature.

The distance from the target to each source is calculated using the displaced version of Equation 4.1. The output of retrieval is a list of domains and their distance to the target. We sort this list on the distance parameter, selecting the nearest sources that lie within some threshold distance from the target. (Initial testing resulted in setting this threshold at a distance of 10 units, but this will be dealt with in the next Chapter). These selected candidate sources are then passed onto the subsequent phases of mapping and validation, which examine in detail their ability to form an analogy with the given source(s).

4.4 Mapping Implementation

Kilaza has been designed in a modular fashion, allowing it to work with a number of alternate mapping models. Mapping was not a direct focus of this work, so the mapping model we describe is relatively standard, being a variant of the Incremental Mapping Model (Keane and Brayshaw, 1988; Keane *et al*, 1994). Our prime requirement for the mapping model is not the inter-domain mapping itself, but the candidate inferences that are mandated by the mapping. However, the mapping must be identified before the candidate inferences are generated.

Like other incremental models, this mapping model is based on two activities known as *root selection* and *root elaboration* (Figure 4.10). Traditional root selection is based on identifying “root predicates” which are

typically the controlling causal relations in a domain. Root selection identifies a correspondence between two selected "root" predicates and the mapping between these predicates is known as a root-mapping. Root elaboration then identifies each of the consequent mappings that are implied by the root mapping, mapping each of the argument pairs in turn. Mapping proceeds as a sequence of root-selection and root-elaboration activities, gradually building up a single inter-domain mapping. Firstly we shall examine the root selection mechanism used in this model, and then the root elaboration process.

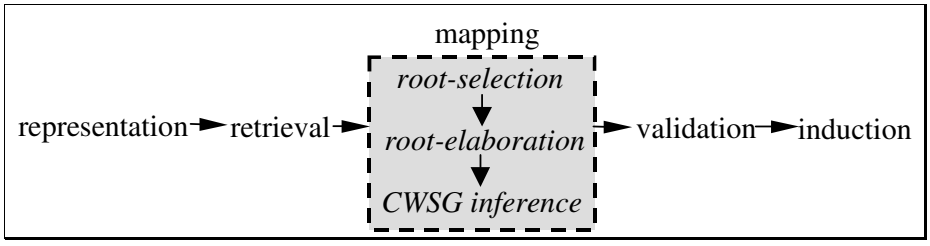

Figure 4. 10 – *Details of the Mapping Processes*

4.4.1 Root Selection

Source domains generally have more high-level causal relations than the target, (generally) resulting in a smaller number of root predicates. We extend the focus on structure that was introduced in the retrieval model, by using another structural feature in the root selection process. The objective of this is to identify predicates at the same hierarchical level within the two domains, and use these to form the root mapping.

The root selection process examines the "Order" of predicates within the domain description, as illustrated in Figure 4.11. Objects are defined as order zero and first-order relations that connect two objects are defined as being of order one. The order of a causal relation is defined as one plus the maximum order of its arguments. Calculating the order of entities is a straightforward process, which focuses on the argument structure of each domain.

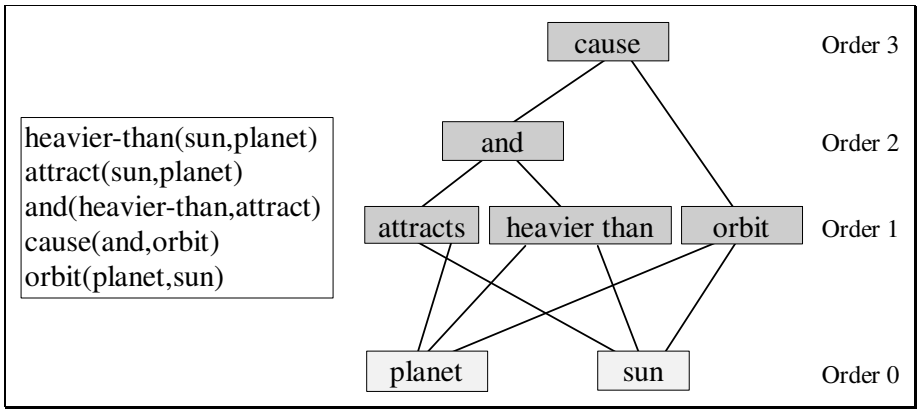

Figure 4. 11 - *Mapping Identifies the Level of Each Predicate in the Domain*

Root selection identifies the highest order predicate(s) in the target, and then identifies all predicates of the same order from the source domain. If multiple root-predicates are available for selection, then Kilaza will favour a mapping between identical relations. However, predicate identicality in Kilaza is a preference rather than a hard constraint.

The *atom* target identifies two Order-1 predicates that can participate in a root mapping, namely; `heavier-than` and `attracts`. Because these are the highest order relations in the target domain, the mapping will be grown from these predicates. The relations in the *solar-system* source that are of the same order are; `heavier-than`, `attracts` and `orbits`. These predicates are selected to participate in root mapping(s) with the identified target predicates. Root selection favours mappings between identical relations before proceeding with non-identical relations.

One other feature of this structurally sensitive algorithm for selecting "root mappings", is that it uses different structural features than those used during the retrieval phase. This design decision was seen as essential, to ensure we were not simply developing only those analogies that are preferred by the retrieval process.

4.4.2 Root Elaboration

Root elaboration extends each root-mapping, placing the corresponding arguments of these relations in alignment. If these arguments are themselves relations, then their arguments are mapped in turn and so on until object arguments are mapped. These subsequent mapping activities contribute to the same global inter-domain correspondence, and so none of these mappings may violate the 1-to-1 constraint. So, before accepting any root mapping, Kilaza tests to see if the entire root-mapping is consistent with the current inter-domain comparison. Only when this test succeeds does the root elaboration process proceed in earnest.

So for the *solar-system:atom* example, the mapping between the `heavier-than` relations in the two domains is elaborated to map the `nucleus` with the `sun` and the `electron` with the `planet`. The next root mapping aligns the two `attracts` relations, and this mapping relies on the same object mappings. Thus, all target items are incorporated into the inter-domain mapping, accounting for all target information

4.4.3 CWSG - Copy With Substitution and Generation

The crucial factor in finding novel and useful analogies, is that the mapping allows us identify the set of candidate inferences. The "pattern completion" algorithm for inference generation is generally referred to as CWSG - Copy with Substitution and Generation (Holyoak and Melz, 1994; Markman, 1997). Unmatched source structures that participate in the mapping are identified as inferences, after being suitably substituted by their corresponding items from the inter-domain correspondence (Figure 4.12). Unmatched source elements are added to the inferences in the form of Skolem objects or relations.

Unlike the standard CWSG algorithm however, we do not immediately add these candidate inferences in the target domain. So unlike the standard CWSG algorithm, inference generation is not an immediate process that is blind to the candidate inferences themselves. The inferences that are generated by the mapping phase must first be validated by the validation phase, before the mappings and its inferences are accepted. It is the

validation phase that must detect invalid analogical inferences and protect the integrity of the target description.

The mandated inferences are generated and passed to the validation phase where they are considered for inclusion in the target domain. Keane (1996) identifies the *adaptability* of the candidate inferences as a distinct influence (along with structural, semantic and pragmatic factors) on analogical mapping. Keane defines adaptability as the usefulness of source information to a target problem, so sources that contain more adaptable information are preferred over those that are less adaptable. For example, a source that uses only those objects already contained with the target domain will be considered more adaptable than a source that prompts the use of an additional target object. So if a source for the *atom* problem required the use of some additional object from the *atom* domain, this would be considered less adaptable than a source that did not rely on such an object. Keane (1996) uses the example of an analogy generating an invalid inference, indicating that such an inference has very low adaptability. We see validity as a special case of adaptability, where the candidate inference cannot be adapted to the target domain. The mechanism we propose is a possible explanation for some of the "adaptability" effects that were noted by Keane, as the validation (and adaptation) process identifies how the candidate inferences influence which of the possible inter-domain mappings is finally accepted.

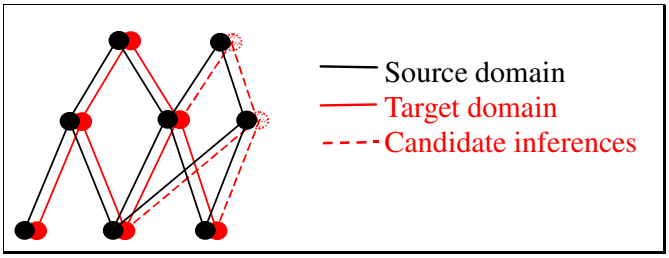

Figure 4. 12 - Structural perspective on CWSG Inference

Another important difference between Kilaza's inference mechanism and the CWSG algorithm, relates to Kilaza's use of instance nodes specific to each domain description. When an item is transferred directly from the

source to the target domain, a new instance node is generated for the target domain and is appropriately connected back to the taxonomy (for future use). For example, consider the source relation *orbits* that is to be added to the *atom* domain by the CWSG algorithm (let us assume that the relevant predicate is successfully validated). So, the source relation:

```
orbits663 planet664 sun665
```

would become something like the following in the target domain:

```
orbits45634 electron364 nucleus36537
```

Note that the `electron364` and `nucleus36537` nodes existed in the original source description, but `orbits45634` is a newly generated instance node that was created during inference generation.

4.5 Validation Model

As stated in Chapter 2, Phineas in the only detailed model of the post-mapping verification/validation process, but Phineas adopts a very domain-specific approach to verification. In this section we present a new model for domain-independent analogical *validation*. The presented validation model is part of the post-mapping phase that also includes the adaptation process. (In Chapter 1 we call this phase "Validation" rather than use Keane's (1994) original "Adaptation" to emphasise our primary focus, but this phase also addresses the adaptation process).

Spiro *et al* (1989) identify two ways in which an analogical comparison can generate invalid expectations about a target problem. First, an *over-extension* of the source analog to the target generates one type of mis-conception. So, an overextension of the *solar-system:atom* analogy might lead to the inference that the nucleus heats the electron. The other concerns *omissions* in the source information, which can also result in misleading expectations. An omission might omit the precondition that the sun is heavier than the planet, possibly resulting in the incorrect inference that the nucleus orbits the electron (rather than the electron orbiting the nucleus). In this thesis, we will focus on detecting invalid *over-extensions* of source domain information and will not look at source *omissions* any further. Thus,

validation in Kilaza focuses on detecting over-extensions and mis-applications of source material to the target problem.

After generating the candidate inferences, Kilaza first assesses their validity before introducing them to the target domain. Validation is particularly important in this model, because the semantically diverse retrieval model might identify sources that contain potentially non-analogous information - and this information must not be introduced into the target domain. The inferences that these comparisons suggest might contain any combination of relations and arguments, not all of which will even be valid. For example, consider an over-extension of the following analogy that might be identified by Kilaza (see Figure 4.13). In this Figure the analogy generates inferences (highlighted) that do not hold in the target domain. We point out that Kilaza is intended to operate on a memory of domain descriptions possibly created by different authors. Thus, the intended meaning of a relation (say) in one domain, may be slightly different from that in another domain. Therefore, no guarantees can be given about the semantic content of the inferences that may arise.

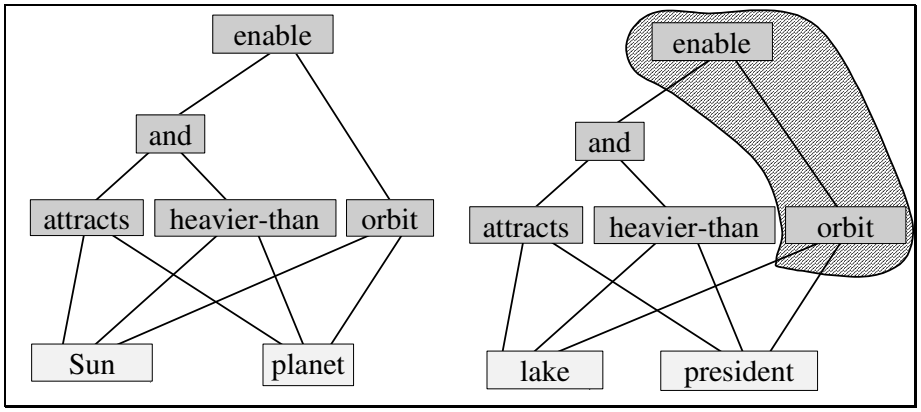

Figure 4. 13 - *Over-extension of the Analogical Comparison*

Kilaza will identify and reject any non-analogous information based on the candidate inferences that are generated. To date, the analogy literature has focused almost exclusively on valid analogical comparisons, and rarely if ever addressed their converse. Non-analogies are of particular interest as they

highlight some of the constraints on inference that usually go unnoticed - particularly when only widely accepted analogies are considered as input to the relevant algorithms.

Validation is broken into two processes of *validation* (per se) and *adaptation*, as depicted in Figure 4.14. *Validation* implements an acceptance filter, while *adaptation* attempts to modify any invalid inferences so that they better fit their target domain. Because the objective behind a particular novel analogy will not be known to Kilaza, *validation* and *adaptation* must operate in the absence of pragmatic factors. These two processes must also operate in a domain-independent manner, and cannot be tied to any one problem area. The objective of our validation mechanism is not to guarantee the correctness of all accepted inferences, as this would require a deep model of every possible target domain. Kilaza's validation mechanism will focus on rejecting clearly invalid inferences, irrespective of the target domain.

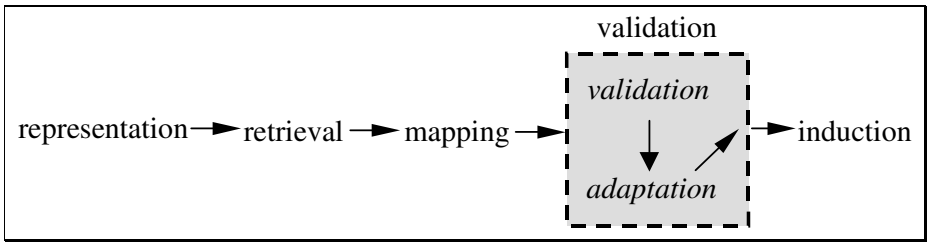

Figure 4. 14 - *A Two-Part Model of Analogical Validation*

Phineas (Falkenhainer, 1988-b) also performs verification by comparing the inferences against other known facts. Phineas operates at the *inter-predicate* level by comparing inferences against known facts and its "behavioural abstractions". Of course, even a complete model of validation may even be insufficient for some inferences, which may require physical verification or may even be un-verifiable. Validating the "heart is like a pump" analogy requires experimentation, while validating the following may be impossible: "the day before the universe was created was like D-day". In fact, validating some of the deeper implications of Rutherford's *Solar-*

system:Atom analogy led to the ultimate rejection of this comparison by astro-physicists[2].

Rather than adopt the inter-predicate approach, Kilaza focuses on a lower level of interaction between the source and target domains referring to it as the *intra-predicate* level. Intra-predicate validation will assesses the validity and integrity of individual predicates, in the absence of general schemata or a wider problem context. It was expected that intra-predicate validation might make use of the taxonomy to support any validation activities. The objective is to remove any dependence between validation and a specific problem domain and allow validation to occur across all problem domains. Validation is achieved by focusing exclusively on intra-predicate constraints, using the memory structures outlined earlier (see Chapter 3) to support both validation and adaptation.

For the purposes of validation, we distinguish between high-order inferences and first-order inferences. Because high-order relations assert a causal connection between two other relations, it is difficult (if not impossible) to validate such inferences without reference to the "antecedent" and "consequent" relations. Validating causal predicates can not even rely on spatio-temporal contiguity (Pazzani, 1991). For example, there is neither spatial nor temporal contiguity between the antecedent and the consequent of the following predicate, yet it is still a valid predicate - `cause(aerosols, ozone-hole)`. Thus, we do not validate these high-order relations. As we shall see shortly, first-order predicates offer more opportunities to identify invalid inferences, and so we focus on validating such inferences.

We identify a number of different modes of validation, which are described in the following sections.

---

[2] When the electrons circle round the nucleus, they are constantly changing their direction. According to classical electrodynamics, such electrons should continuously emit radiation. While doing so, they should lose energy and thus spiral into the nucleus. This means *every atom is unstable*, quite contrary to our observation (Miller, 1996; Wilson and Buffa, 1997).

1) Identical Predicate Validation

2) Partial Predicate Validation

3) Commutative Predicate Validation

4) Functional Feature Validation

5) Adaptation

As we shall see, Kilaza's predicate repository and taxonomy are used in different ways to support the post-mapping activities in the list above.

4.5.1 Identical Predicate Validation

The first three forms of validation are based primarily on the contents of the predicate repository. *Identical predicate validation* is the simplest and most reliable form of validation. It is performed by comparing each candidate inference against an identical predicate found in the predicate repository. This is the strongest form of validation and (effectively) ensures the validity of the inference.

To achieve this mode of validation, Kilaza must identify a previous instance of the candidate inference. The relational predicate in the candidate inferences is used to identify the corresponding generic node in the taxonomy, as it is connected to all instance nodes in the predicate repository (as detailed in Chapter 3). The generic node is used to retrieve all past instances of this relation, and these predicates are checked for the presence of an identical predicate - or else this mode of validation is unsuccessful. The following is read as <*candidate inference*> is validated against <*identical predicate*>;

<*Reln agnt ptnt*> ≈ <*Reln' agnt' ptnt'*>

where *Reln, agnt* and *ptnt* correspond to the corresponding values of the inference's relation, and *Reln' agnt'* and *ptnt'* correspond to some pre-existing identical predicate. So, the candidate inference `orbit(electron, nucleus)` is validated against the `orbit(electron, nucleus)` that is found in the *solar-system* domain.

Identical predicate validation implements intra-predicate constraints by ensuring the inference is as consistent as the contents of the predicate repository. (As pointed out in Chapter 1, Kekule's carbon-ring analogy did not require any new inferences, but applied a new structure or organisation to existing information). However many inferences, and most creative inferences, will not be successfully validated by comparison against known predicates. So additional mechanisms are required to validate previously unseen predicates.

4.5.2 Partial Predicate Validation

If an identical predicate to the candidate inference is not found in memory, then we resort to the next form of validation, which again uses the predicate repository. Partial-predicate validation operates in a "piece-meal" fashion, validating the agent and patient roles of a predicate independently. It validates the *relation-agent* pair separately from the *relation-patient* pair, using two pre-existing predicates identified from the predicate repository.

<Relation agnt ptnt> ≈ *<Relation' agnt' _>* & *<Relation' _ ptnt'>*

where the underscore _ signifies a wildcard term, and & is the logical `and` operator. For example, this mode of validation will validate the following candidate inference.

```
kill(lion man) ≈      kill(lion' zebra')
                      kill(gun' man')
```

Where the underscored items represent don't care terms. Thus, the arguments `zebra'` and `gun'` are effectively ignored by the validation process.

This mode of validation separates the dependencies between the relation and its agent role, from the dependency between relation and its patient role. Clearly, this can result in validating some untrue predicates (below), but it vastly increases the range of predicates that can be validated.

```
double(two eight) ≈   double(two' four')
                      double(four' eight')
```

So partial-predicate validation is a weaker form of validation than identical predicate validation. However, this mode of validation does ensure that the

agent and patient arguments may be validly used with the given relation, either separately or in conjunction with one another.

Partial predicate validation offers two positions that might be adopted for validating inferences. The "strong" position would involve accepting only valid inferences, immediately rejecting any inferences not known to be valid. We adopted the weaker position of rejecting only *clearly* invalid inferences, doing so for the sake of generality. Thus, inferences whose validity is uncertain are considered to be potentially valid and are accepted by validation. This "weak" position allows many novel analogies to be considered that would be rejected by the "strong" position.

4.5.3 Commutative Predicate Validation

If the previous means of validation prove unsuccessful and the relation of the inference is commutative, then another means of validation is available. Examples of commutative relations include `next-to`, `besides` and `looks-like`. Commutative predicates can be found in one of two equivalent forms, because the relation is oblivious to the order in which the arguments are supplied.

`next-to(man house)` ≈ `next-to(house' man')`

As well as being validated against an identical predicate, the inference `next-to(man, house)` may be validated against the predicate `next-to(house, man)`. (Note that one order of these arguments has already been tested during Identical Predicate Validation).

(reln arg1, arg2) ≈ *(reln arg2, arg1).*

If the commutativity of the inferred relation is unknown it is assumed to be non-commutative, as the great majority of relations are not commutative. This additional mode of validation is not available to unknown or non-commutative inferences.

Commutative Partial Predicate Validation

Commutative relations may also need to be validated in a piece-meal fashion. In this case the agent role can be validated against either agent or patient positions - in isolation from one another, So the predicate

```
next-to(man house) ≈ next-to(man' car')
                     next-to(house oak-tree')
```

or alternatively

```
next-to(man house) ≈ next-to(car' man')
                     next-to(oak-tree' house)
```

Here, the patient role can also be validated against either agent or patient positions of a different instance of that relation-argument pair.

<Reln a b> ≈ (<Reln a _> OR <Reln _ a>) AND
(<Reln _ b> OR <Reln b _>)

This introduces an additional means of validating the relevant inference, and further widens the scope of the inferences that may be validated.

4.5.4 Functional Feature Validation

When an appropriate predicate (or predicates) cannot be found to help validate an inference, we resort to the use of functional features. This is the most general form of validation used by Kilaza, allowing validation even when there are no recorded instances of the predicate concerned. Therefore, this from of validation is particularly suited to dealing with the novel inferences that sometimes result from creative analogies.

Functional-feature validation rejects inferences when they violate the relation restrictions. Functional-feature validation focuses on the attributes required by the filler of each argument role in a given relation. This mode of validation is directly supported by the functional features described in the Chapter Three. Two regions of the taxonomy are directly involved in this validation process. The first-order predicate hierarchy identifies the functional features required by the two arguments. Secondly, the object hierarchy is used to retrieve the features that are used to describe the

arguments. Each object argument must either directly possess the required functional features, or possess one of the features subtypes.

There are two reasons to suspect that these *selection restrictions* (Jurafsky and Martin, 2000) might be usefully applied to analogical inferences. Firstly, analogies (and their inferences) are based on semantically "deep" comparisons, and this depth even serves to create generalised schemata (Gick and Holyoak, 1983). Thus, the inferences that are generated by analogies (even creative analogies) are based on this "deep" inter-domain similarity. Therefore, we might expect these selection restrictions to be more successful than when validating predicates not founded upon an analogical comparison.

Secondly, the restrictions used by validation may be subsequently re-used by the following adaptation process. This adaptation process combines the selection restriction for arguments, with a taxonomic restriction on the relation to identify the *adapted* predicate. Thus, the selection restriction is also used to support analogical adaptation. These two factors combined with the generality of selection restrictions, led us to use a feature-based selection restriction to support validation and adaptation.

Kilaza treats the functional features of a relation's role as one set, and treat the features of the argument objects as another set. Functional feature validation ensures the following constraints are satisfied:

$$FA\text{-}agnt - OA\text{-}agnt = \phi$$

$$FA\text{-}ptnt - OA\text{-}ptnt = \phi$$

Where; *FA-agnt* and *FA-ptnt* are the functional features of the agent roles respectively, and *OA-agnt* and *OA-ptnt* are the object features of the agent and patient arguments respectively, and the "-" is the set-difference operator. In other words, the functional features less the object's features is the null set, ensuring the arguments possess all functional features. If either agent or patient restrictions do not hold, then the inference is rejected.

Validation Summary

The different mechanisms used to perform validation carry different levels of confidence on the inference's validity. We distinguish between three different levels of validation, based on which of the above mechanisms were used to validate the candidate inference.

- *Validation Mode 1 - Identical Predicate Validation.*

The inference is equally as valid as the contents of the predicate repository. These inferences are effectively guaranteed to be valid.

- *Validation Mode 2 - Partial Predicate Validation, Commutative Predicate Validation* and *Commutative Partial Predicate Validation*

The predicate is accepted as probably valid, based on a pair-wise comparison of its argument roles to pre-existing predicates. That is, the agent seems a valid agent argument and the patient argument seems like a valid patient argument. However such predicates are not guaranteed to be valid.

- *Validation Mode 3 - Functional Feature Validation*

The predicate is accepted as potentially valid, based on a more abstract comparison to a similarity template. These predicates are expected to be possibly valid, but there is no guarantee that all uses of the relevant predicate will be covered by the functional feature definition of the predicate. However, it is expected that this mode of validation will validate more novel predicates than Validation Mode 2.

4.6 Adaptation

The validation process identifies invalid predicates from the candidate inference set - but not all invalid inferences must be completely discarded. Some of the inferences may contain recoverable information, requiring only a minor modification to turn an invalid inference into a valid one. These adaptations can preserve much of the essential content of the original inference and allow appropriate elaboration of the target problem.

A very significant factor guiding adaptation, is the pragmatic role that an inference must play within the target domain. That is, adaptation normally requires information on the analogisers "goal" as well as access to other

information beyond the scope of the analogy (Holyoak, Novick and Melz, 1994). However, Kilaza does not have access to such information, as it is merely trying to discover valid analogies independent of a specific problem context. Thus, adaptation in Kilaza is going to be limited in what it can achieve.

Kilaza focuses on predicates consisting of a transferred relation, plus at least two target-originating objects. This constraint ensures that modifications are only made to relations that originate in the source domain.

4.6.1 Relation Adaptation

Adapting the relation of an inference uses two pieces of information to identify an alternative relation (if such a relation exists). Adapting relations is a three-step process, as depicted in Figure 4.15. Firstly, the taxonomy is used to identify similar relations to the rejected one, which might be applicable in the target. The second and third step use the functional features in the reverse manner to their use in validation - during adaptation the functional features are usd to identify potentially useful relations. The second step of adaptation uses the functional features of the rejected relation as a selection constraint on the relations identified in the first step. In the final step of adaptation, the features of the arguments to the rejected inference are used as a selection constraint on these relations.

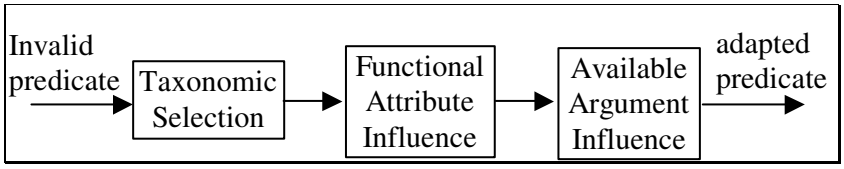

Figure 4. 15 - *Diagram of Adaptation in Kilaza*

Adaptation in Kilaza considers as adaptations, the super-ordinate relation to the rejected relation, plus all predicates below this node. Thus, the alternate relations include the super-ordinate, the sibling relations and their children. This taxonomic constraint ensures that the adapted relation is semantically

similar to the rejected relation - and thus should be able to play the role of the original relation within the target description.

Source:
```
Own (man, car)
Reach (man, destination)
drive (man, car)
Enable (drive, reach)
```

Target:
```
Own (man, horse)
Reach (man, destination)
```

Figure 4. 16 - *Sample Analogy requiring inference validation*

For example, consider the analogy between driving a car and riding a horse shown in Figure 4.16. The first-order inference requiring adaptation is:

```
*drive (man, horse)
```

This inference is ultimately rejected because `drive` requires a patient argument that is mechanical. Adaptation begins by identifying that `operate-control` is the super-class of `drive` (see Figure 4.17). The other relations within this class include `flies`, `sail`, `paddle` and `ride`. These relations are identified as the initial set of potential adaptations for the rejected relation (there are no subclass to any of these relations).

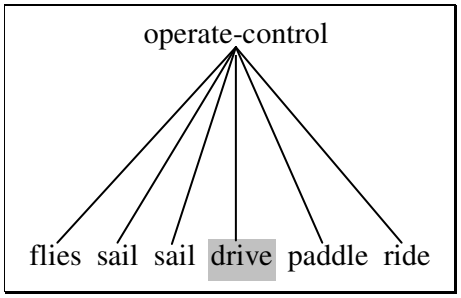

Figure 4. 17 - *Adapting the "sail" relation*

Next, we instigate a spreading activation process to refine this initial selection. This originates in the functional features of the rejected relation, identifying relations with the most similar argument restrictions. So our example would identify relations with similar functional features, including: `ride`, `fly`, `sail` and `paddle`.

The final step uses a spreading activation process originating in the arguments of the invalid inference. This identifies relations that require the available objects. So in the current example, the agent argument will contribute equally to the relations identified earlier in the process. However, the patient argument (`horse`) will contribute to the `ride` relation, because it best matches with its features. (The agent features remain unsatisfied for the alternative relations, including `fly` and `sail`).

These three activities will identify a number of possible relations, all of which are sorted based on their activation levels. Kilaza now validates each of the possible adaptations in turn, in order of their activation level. The functional features of each relation in turn are tested against the available arguments. The first relation to be successfully validated is then chosen as the accepted adaptation - or if none of the alternatives are successful then, adaptation fails (and the predicate is rejected without adaptation).

So, the adaptation process will suggest `ride` as the most appropriate adaptation for the `drive` relation in the invalid predicate, generating the following inference.

```
ride (man, horse)
```

This inference is accepted by the validation process (Note: Kilaza does not recursively allow the adaptation of a failed adaptation). Combining the taxonomic restriction with the functional feature restriction ensures that the adaptation process does not merely generalise the offending relation to its super-ordinate relation. Generalisation can involve significant information loss, and generate ineffective predicates. Of course, the super-class of the rejected relation is among the alternatives considered, but it is *not* the *only* alternative considered.

4.6.2 Argument Adaptation

In contrast to relation adaptation, adapting the arguments of relations is a much more limited activity. The between-domains nature of analogical comparisons, where the objects of the source and target originate in very different domains, means that the objects of one domain are quiet different to those in the other. Thus, there is often little obvious similarity between the mapped objects in the two domains. Consider for example the *atom:solar-system* analogy juxtaposing a `nucleus` with the `sun` and an `electron` with a `planet`; or Kekulé's *snake:carbon-chain* analogy juxtaposing the `carbon-chain` with the `snake's-body`. In this case, adapting one object based on its expected similarity to an object in the other domain would not appear to be a particularly useful approach. We now describe the situations in which argument adaptation is undertaken, and the means by which it is performed.

Argument Identification and Skolem objects

We only consider adapting arguments when these arguments are transferred directly from the source to the target domain. However, because of the between-domains nature of analogy, we do not introduce skolem objects into the target domain, as is traditionally performed by the CWSG algorithm (Holyoak, Novick and Melz, 1994). Instead, we first examine the target domain for any unmapped objects that may fill the role of the additional source material. Only if this process is unsuccessful do we consider creating skolem objects in the target domain.

Many of the partial predicates generated by CWSG can be completed by reference to unmapped target entities (when they exist), including any appropriate partial predicates containing unmapped objects (see Section 3.3). Again, the functional features help to identify appropriate objects from the collection of unmapped target objects.

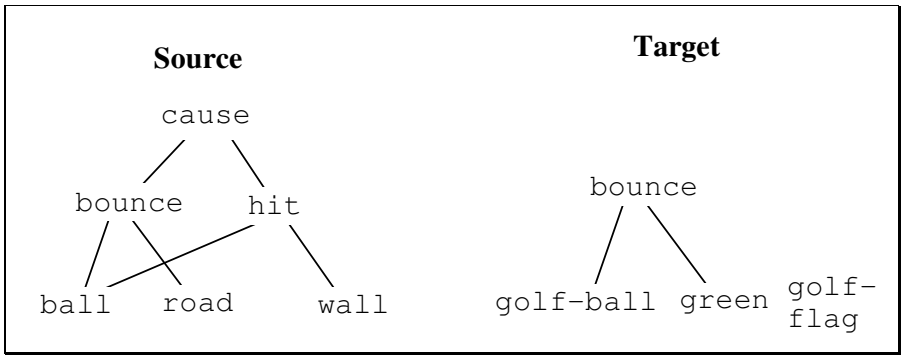

Figure 4. 18 – *The Target Contains some "unused" Objects*

For example, consider the source domain described in Figure 4.18 above. Now consider the target domain consisting of just one predicate:

```
bounce(golf-ball, green)
```

plus one isolated object reference `golf-flag`, in the form of a partial predicate.

```
(golf-flag)
```

The candidate inference created by CWS (Holyoak *et al*, 1994) but before generating the missing items will be:

```
hit(golf-ball, nil)
```

Kilaza now searches for a suitable argument in the target domain to complete this predicate. First, it identifies all unmatched objects in the target, as they might fill the unoccupied argument position of the candidate inference. This process will identify `golf-flag` as an unmatched object. (A more realistic memory might be able to access information not explicitly specified in the target domain, using this to complete the mapping).

Next Kilaza uses the functional features of the inferred relation to select from among the unmatched target objects. Only objects that satisfy all functional features are considered plausible fillers for the empty argument position. Subsequently, functional features will confirm that this is a plausible argument for this relation, and so the following predicate is generated:

```
hit(golf-ball, golf-flag)
```

So functional features also assist in the task of elaborating inferences, as well as validating and adapting them.

4.7 Conclusion

The overall objective of this project was to create a model of analogy that was capable of generating novel analogical comparisons for some given target problem. In pursuit of this goal, we presented our model of analogical reasoning that encompasses the phases of *retrieval, mapping* and *validation*. This multi-phase model was built on top of a two-part memory encompassing a taxonomy and a predicate repository (as described in Chapter 3).

The *retrieval* model aimed to overcome the limitations of retrieving semantically similar sources, which are associated with existing retrieval models. Instead Kilaza focused on the expected structural similarity between the two domains of a viable analogy, and used this as a basis for retrieval. Structural features are used to describe the graph-structure of each domain, and these features are combined to form a structure space. Structure based retrieval is then performed within this structure space. The semantic-free nature of this space means that retrieval can identify the semantically diverse candidate sources, that are associated with novel and creative analogies. The creativity of our model is partly founded on its ability to identify semantically distant sources that are, at least structurally capable of forming an analogy and have the structural potential to supplying inferences to a given target. A threshold distance is used within structure space to identify the candidate sources.

Candidate sources are passed to the *mapping* phase, where the inter-domain correspondence is identified using a modified version of the incremental mapping model. This uses a structurally sensitive root-selection process to map predicates at the same hierarchical level. Because of the semantically diverse sources that Kilaza must consider, it uses predicate identicality as a preference - not a hard constraint. This allows the exploration of many more analogies than would have been possible were identicality enforced as a hard constraint. The root-elaboration process compiles the final

inter-domain mapping from several of these root mappings. Inferences are generated by the usual "pattern completion" model, but all inferences are passed to validation to determine their validity.

Finally, we described the *validation* model that encompasses validation and adaptation processes. This validation model operates in any target domain, and does not rely on pragmatic factors. Validation focuses on the acceptability of individual predicates by firstly comparing them against similar predicates in memory. Novel predicates are validated using the functional features that restrict the arguments supplied to defined first-order predicates. Kilaza attempts to adapt all invalid inferences, before they are rejected. We described how the taxonomy and the functional features are used together to adapt invalid inferences, so that they better fit their target arguments. We also described how missing target objects may be identified, again using these functional features.

Having described the model and each of its phases, the next chapter will focus on assessing the abilities and limitations of this multi-phase model for discovering novel analogies.

Chapter 5

Tests of Structural Retrieval

"Though analogy is often misleading, it is the least misleading thing we have."

- - Samuel Butler, Notebooks, Life. Music, Pictures and Books. 1912

"Computers are useless. They can only give you answers."

- - Pablo Picasso

5.1 Introduction

We have seen a description of the Kilaza model, detailing *how* each phase of the model operates. In this chapter and the next, we will assess *how well* each of these phase-models contributes to the goal of developing a model of analogical creativity. Our assessment of these phase-models is divided into two result chapters. This first chapter will focus on results produced by the retrieval model, while the next chapter will focus on the validation results.

In this chapter we will assess the effectiveness of the retrieval model, and its ability to identify source domains with creative potential. Assessing the creativity of retrieval will encompass two main parts. First we will assess how many of the identified sources form a useful mapping with the given target, and secondly, we will assess how many of these sources can supply inferences to that target. Additionally, we will examine whether Kilaza's retrieval model overcomes the semantic bias that is associated with other retrieval models (as discussed in Chapter 2).

We will evaluate Kilaza's retrieval model by examining its operation on two different collections of domains. The first collection has 14 domains, each containing from 10 to over 100 predicates. This collection is called the Professions knowledge-base and was developed by Veale (1995). (These domains were originally developed as metaphoric comparisons, but we use their latent ability to form analogies). The second collection contains 81 domains, each with between 1 and 15 predicates. This collection was compiled by the author specifically for this project. Significantly, both knowledge bases were developed without reference to the objectives of the project. The second collection was inspired by domains found in the analogy literature, including: *solar-system:atom, heat-flow:water-flow and tumour:fortress* (Duncker, 1945, Gentner, 1983; Falkenhainer *et al*, 1989), but contains an assortment of other domains yielding its name, the Assorted Knowledge-Base (KB).

5.1.1 Structure of this Chapter

This chapter starts with a detailed description of the two KB's that are used throughout this and the next chapter. We follow this with a brief overview of the entire model of creative analogising, encompassing the phases of retrieval, mapping and validation. We describe how the Kilaza model generates Rutherford's *solar-system:atom* analogy. We examine the retrieval model on the two collections, the Professions KB and the Assorted KB. The objective of this chapter is to see whether the retrieval model does indeed find useful and productive analogies. We examine how the parameters that measure retrieval,

correlate with the mappings and inferences that were generated by each of these analogies. Then we assess the implications of these results on our creativity model. The chapter concludes with an overall assessment of the usefulness of structure-based retrieval for finding novel analogies.

5.2 The Test Suites

5.2.1 The Professions KB

Before we look at the results, we must look at the domains supporting the testing that produced these results. The first collection of domains consists of descriptions of fourteen professions, including *accountant, butcher, priest* and *scientist* (See Appendix B). These descriptions were created by Veale (1995) and range in size from 10 to 105 predicates (M=55.4, SD=29.3).

The original domain descriptions included attribute information that described the objects in each domain. These attributes were not developed around a taxonomic structure that was compatible with that of Kilaza, and so could not be modified to be totally compatible with it. Furthermore, unlike the Sapper model (Veale, 1995), Kilaza does not perform attribute matching. However, similar attribute information was made available by describing each object with appropriate attributes from Kilaza's taxonomy.

One important feature of the Professions KB is its reliance on many different instances of just a small number of relational predicates, including `control`, `affect`, `depend`, and `part`. The domains range from using just 6 distinct relational predicates (ignoring duplicates) to the most diverse domain that uses 15 (M=8.9, SD=2.2). Another important feature is that the Professions KB does not make use of a set of clearly identifiable high-order relations (such as a `cause`, `result-in` or `inhibit` relations between two first-order predicates). Figure 5.1 contains part of the "Butcher" domain, and illustrates the repeated use of a small number of relational predicates. All these features have a direct impact on Kilaza's retrieval results, and on the results of the mapping and validation models. (These differences will be discussed in

detail in section 5.4.3). Note also that all concepts are defined in terms of a predefined categories, and these categories are not inspired by any specific problem-solving goal (Barsalou, 1983).

```
(CREATE-NEW-FRAME  BUTCHER
(super analog)
(predicates
     (DEPEND       PERSON       PERSONAL-HEALTH)
     (PART         FAMILY       FAMILY-RELATIVE)
     (PART         GENE-POOL    CHARACTERISTIC)
     (PART         GENE-POOL    GENE)
     (DEPEND       FAMILY       GENE-POOL)
     (PART         FAMILY-TREE  FAMILY-RELATIVE)
     (DEPEND       FAMILY-TREE  FAMILY-BREEDING)
     (DEPEND       FAMILY       FAMILY-TREE)
     (DEPEND       PERSON       FAMILY)
     (LOCATION-OF  BUTCHER      ABATOIRE)
     (AFFECT       BUTCHER      LIVESTOCK)
      …
```

Figure 5. 1 - *Part of the "Butcher" Domain from the Professions KB*

5.2.2 The Assorted KB

The second collection of domains used in testing, the Assorted KB, includes many of the frequently referenced domains in the analogy literature; including the *solar-system, atom, heat-flow* and *water-flow* domains. It also includes an assortment of other domains describing *golf, soccer, story-telling,* and *requited-love* (see Figure 5.2). The 81 domains of the Assorted KB use 108 distinct (*ie* non-repeated) relations while the Professions KB uses only 16 distinct relations. The Assorted KB also uses a distinct set of high-order relations connecting other predicates (including `cause`, `and`, and `inhibit`). Each of the Assorted domains contain between 1 and 15 predicates (M=4.16, SD = 2.9). The average number of distinct relational predicates in each domain is M=3.48, indicating

that most relational predicates are used just once in each of the Assorted domains.

So, the Professions KB has large domains described by repeated sets of very general relational predicates, while the Assorted KB has small domains that mostly use just a single instance of a very specific relation. Therefore, these collections present very different challenges to Kilaza's retrieval and other phase-models. The distinct sets will later be used for testing purposes.

```
(CREATE-NEW-FRANE unrequited-love
(predicates
    (loves        tom    mary)
    (loves        mary   joe)
    (loves        joe    mary)
    (jealous-of   tom    joe)
    (cause        loves  jealous-of) ))
```

Figure 5. 2 - *The Requited-Love Domain from the Assorted KB*

5.3 Overview of Kilaza and The *Atom:Solar-system* Analogy

Before we present the detailed results generated by the retrieval model, we will first see how Kilaza finds Rutherfords' *solar-system:atom* analogy. This example illustrates how Kilaza's components combine to generate Rutherford's famous analogy. This overview will encompass the phases of *retrieval, mapping* and *validation*. However, because none of the inferences in this example are rejected, we will only provide a cursory description of the validation process.

5.3.1 Discovering Rutherford's Analogy

Before Rutherford's analogy was found, the *atom* domain was poorly understood. It is typically depicted as having few first-order relations and no causal structure (see Figure 5.3). Comparing this target to the *solar-system* domain provided the key to understanding the structure of the *atom*.

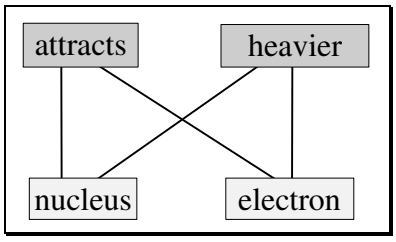

Figure 5. 3 - *The "Atom" Target Domain*

Structural Attribute	Structural Index	Displacement Vector	Locus of Retrieval
Number of object references	4	1	5
Number of unique objects	2	0	2
Number of predicate references	2	2	4
Number of unique predicates	2	0	2
Number of root predicates	2	-1	1
Maximum number of references to object	2	0	2
Number of high-order predicates	0	1	1

Table 5. 1 – *Unmodified and Modified Structural Attributes of the "Atom"*

Now, let us assume that our background knowledge consists of a large number of candidate source domains. From these we must identify a potential source of creative insight for the *atom* target. All candidate sources are mapped into structure space, by examining the structure of each domain description. Among these candidate sources is the *solar-system* domain, whose structural index is (6 2 9 5 1 3 2).

Kilaza then begins the retrieval process in earnest, by interpreting the structure of the *atom* target. The atom's location is derived directly from the

representation in Figure 5.3, giving the values listed in the "Structural Index" column of Table 5.1.

However, we are searching for source domains containing additional transferable material, as these may give rise to the desired inferences. These sources can be found by modifying the "Structural Index" of the target, as described in Section 4.3.3. Kilaza uses a simple heuristic to transform the target's "Structural Index", into its *locus-of-retrieval* by adding the following vector onto the target's structural index: (1 0 2 0 -1 0 1). The *locus-of-retrieval* for the atom target is listed in the last column of Table 5.1. (The justification for applying this heuristic within structure space is given in Section 4.3.3 and Table 4.1 contains examples of this heuristics influence on retrieval).

Adding this displacement vector to the Structural Index, means that the distance between any target and its locus-of-retrieval is $\sqrt{1^2 + 0^2 + 2^2 + 0^2 + (-1)^2 + 0^2 + 1^2} = \sqrt{1+4+1+1} = \sqrt{7} = 2.645$. As we can see from Table 5.2, this displacement reduces the Euclidean distance from the target to the desired sources, by the same distance of 2.645.

The distance in structure space from the *solar-system* to the un-modified structural index of the *atom* is 8, as shown in Table 5.2. However the distance between the *solar-system* and the *atom*'s *locus-of-retrieval* is only 6.08. So using the *locus-of-retrieval* to identify the source domain, means that this source can be retrieved more readily.

The distance between the *atom*'s locus of retrieval to the other domains varies from 1.73 to 29.9 in the Assorted KB (a selection of these distances are listed in Table 5.3). But only 10 of these 81 candidate sources supply any valid inferences to the *atom* target. Significantly, all productive sources were located within a distance of 10 from the locus of retrieval – which is less than 1/3 of the maximum distance recorded. Table 5.3 summarises the results from this sample case, detailing; the distance from the *atom*'s locus of retrieval to each candidate source, the number of inferences generated by that source, and the actual inferences themselves. (Appendix C has full details of all retrieval episodes).

	Structural Index	Locus of Retrieval
Target (atom)	(4 2 2 2 2 2 0)	(5 2 4 2 1 2 1)
Source (solar-system)	(6 2 9 5 1 3 2)	(6 2 9 5 1 3 2)
Difference between vectors	(2 0 7 3 1 1 0)	(1 0 5 3 0 1 1)
Inter-domain Distance	$\sqrt{64} = 8$	$\sqrt{37} = 6.08$

Table 5. 2 – *The "Locus of Retrieval" is Closer to the Desired Source than the Target*

The results in Table 5.3 were recorded after all validation and adaptation processes were performed. As we can see, the *Apple* source domain resulted in the inference (`has-part nucleus electron`) which was validated. However, we will not discuss the details of analogical validation until the next chapter.

Domain Name	Distance in Structure Space	Number of Inferences	Inferences
Throw-Ball	1.73205080	1	`(enable heavier attracts)`
Story-Tell	1.73205080	1	`(enable heavier attracts)`
Cycling	1.73205080	1	`(facilitate attract heavier)`
Burn-Paper	1.73205080	1	`(cause heavier attracts)`
Solar-System-Falkenhainer	3.31662479	2	`(revolves electron nucleus)` `(cause heavier revolves)`
Driving	3.46410161	1	`(facilitate attracts heavier)`
Atom-Falkenhainer	4.89897948	1	`(opposite-sign nucleus electron)`
Apple	5.56776436	1	`(has-part nucleus electron)`
Solar-System	6.08276253	3	`(revolves electron nucleus)` `(and heavier attracts)` `(cause and revolves)`
Sun	9.64365076	3	`(revolves electron nucleus)` `(and heavier attracts)` `(cause heavier attracts)`

Table 5. 3 - *Selected Results of the Atom Retrieval Episode*

Only 3 of the 10 domains listed in Figure 5.3 generate more than one inference. These include the *Solar-system* domain, plus two representational variants of it called the *Solar-system-Falkenhainer* and *Sun* domains. In fact the only difference between the *Solar-system* and *Solar-system-Falkenhainer* domains is that the latter does not have the high-order predicate (`and heavier attracts`). Also, the *Sun* differs from the *Solar-system* domain in that it includes the additional predicate (`enable oxygen-atmosphere habitation`). Each of these domains describe the *solar-system* in different ways, and consequently have different structural indices. Interestingly, all three sources provide the desired inferences to the *atom* domain to complete Rutherford's analogy, and all three were located within a distance of 10 from the *atom's* locus of retrieval (we will return to this fact at the end of the chapter).

None of the three variants of the *Solar-system* source were in fact closest to the *Atom* target. This is not really surprising, given the unpredictable nature of any search for analogies that are novel to the model. We point out however, that the three sources generating the most inferences were located within a distance of 10 units from the target in structure space when the maximum distance was 29.9. However, little can be inferred from this one example. The remainder of this chapter will investigate whether there is any relationship between structure based retrieval and the number of inferences that are generated.

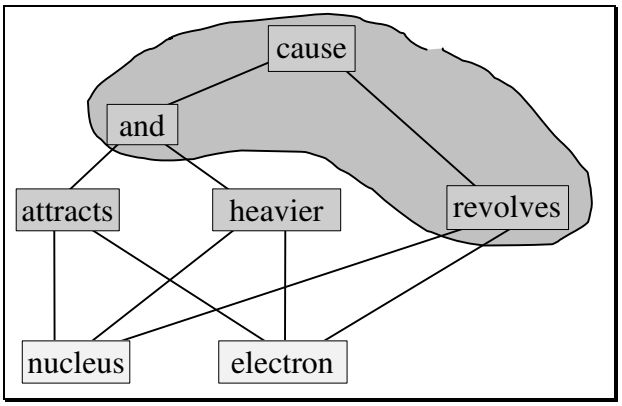

Figure 5. 4 - *Inferences Complete the "Atom" Target*

This example illustrates how structure-based retrieval can identify appropriate source domains without using the semantics of the target. The three sources illustrate that different representational variants of the same information can make the corresponding source either easier of more difficult to identify. Even "re-representing" (Yan *et al*, 2003) the target relation `more-massive-than` as something like `bigger-than`, `larger-than` or `outweighs` will not affect structure based retrieval. This makes it a powerful and resilient retrieval process. However, structure-based retrieval would be affected if the single predicate (`more-massive-than sun planet`) were re-represented as the pair of relations (`larger-than sun planet`) and (`heavier-than sun planet`), because this would impact on the domains structure. We will return to the topic of representation and re-representation in the next chapter.

5.4 Assessing Structure-Based Retrieval

Having seen how the retrieval mechanism works in theory, we now evaluate its performance on the two problem sets. As discussed above, the Professions and Assorted KBs provide different challenges to the retrieval process. Thus, we begin by evaluating retrieval performance on each collection separately. The objective behind testing was to determine if retrieval did actually find sources that generate large mappings and result in many inferences.

Testing the retrieval model followed a leave-one-out strategy. Each domain was taken in turn from the database and used as the target domain. Structure based retrieval was then performed on all other domains in the KB. For each of the resulting analogies, we recorded the number of mappings that were generated, as well as the number of valid inferences that resulted.

5.4.1 Correlations in the Professions Database

Retrieval Distances

The structural distance between each pair of domains from the Professions KB is listed in Table 5.4. The target domains are listed across the top while the source

domains are listed down the left-hand side. Each of the 196 table entry depicts the distance in structure space from that target's *locus-of-retrieval* to the corresponding source. In these retrieval tests we did not apply a hard constraint within structure space to determine successful retrieval. So all sources and their distance to the locus-of-retrieval are reported in the following section. (At the end of this chapter we will perform a further series of tests, applying a hard constraint to structure space, to determine successful retrieval).

These inter-domain distances vary from 2.645 to 230 (M= 80, SD=57.3). As can be seen from the table, it is not quite symmetric because of the displacement between each target and it's locus-of-retrieval. The distance between each target's locus of retrieval and the same source domain is rounded from 2.645 to 3 in this table (see the main diagonal of Table 5.4).

Target Domains

Source Dinaubs / Structural Distance	Accountant	Architect	Author	Butcher	Chef	Composer	Criminal	General	Hacker	Magician	Politician	Priest	Scientist	Sculptor
Accountant	3	68	187	11	63	80	47	188	15	164	122	125	48	42
Architect	66	3	124	56	10	17	21	124	52	98	56	59	21	109
Author	184	121	3	175	124	107	141	14	171	42	69	67	139	226
Butcher	9	58	177	3	54	70	37	178	6	155	112	116	39	52
Chef	61	10	126	51	3	18	16	127	48	103	60	63	15	104
Composer	77	15	109	68	16	3	33	110	64	86	43	47	31	120
Criminal	44	23	143	35	19	36	3	144	31	119	77	80	7	87
General	185	121	13	176	124	107	141	3	172	34	67	65	139	228
Hacker	13	55	174	4	50	66	33	175	3	151	108	112	35	55
Magician	162	96	44	152	100	84	117	36	149	3	43	41	116	205
Politician	119	53	72	110	57	41	74	70	106	45	3	7	73	162
Priest	123	57	70	113	61	45	78	67	110	43	6	3	76	166
Scientist	46	23	141	36	17	34	6	142	33	118	75	79	3	89
Sculptor	44	111	229	54	106	123	90	230	58	207	165	168	91	3

Table 5. 4 – *The Structural Distance between Professions Domains*

Looking more closely at Table 5.4, we can see there is (almost) symmetry between the top right and bottom left halves of results in Table 5.4. If this table reported the distance between domains in structure space, then the table would be perfectly symmetrical. However, instead of using the target's actual location, it is based on the targets modified locus of retrieval (as described in section 5.3.2 above). Thus the distance from target A to source B is not exactly the same as the distance from source A to target B. For example, the *Architect* target is a distance of 68 from the *Accountant* source, but the *Accountant* target is a distance of 66 from the *Architect* source.

Mapping Correlation

The size of the mapping that was generated by each of the Professions analogies is listed in Table 5.5. This measures the number of complete predicates (including arguments) that participate in the inter-domain mapping, varying from 1 to 98 (M=36.3, SD=21.5).

The correlation between the structure space distance and the number of mappings was calculated using Pearsons product-moment correlation and was found to be low and reliable $r\ (196) = -0.331$, $p < 0.0005$. The fact that the correlation is negative indicates that the retrieval metric places useful sources close to the target's locus of retrieval and these domains tend to generate larger mappings. This was the hoped for result and provides support for the role of domain structure in analogical retrieval. Conversely, domains located far from the locus of retrieval tend to generate smaller inter-domain mappings (see Figure 5.5). The correlation is relatively weak (as expected) because of two main factors. Firstly, structural attributes merely approximate the structure of the relevant domains, but do not identify every possible structural feature. Thus, a few structurally dissimilar domains may be located artificially close to one-another because their dissimilarity is not captured by the structural attributes. Secondly, the semantics of the domain descriptions also influences the size of the mapping that can be created, because of the 1-to-1 constraint.

#Mapping Relations	Accountant	Architect	Author	Butcher	Chef	Composer	Criminal	General	Hacker	Magician	Politician	Priest	Scientist	Sculptor
Accountant	24	25	35	23	23	22	22	21	25	19	23	22	25	5
Architect	21	50	84	24	42	52	40	68	28	32	46	47	40	4
Author	22	47	90	28	47	55	41	57	28	33	48	30	41	5
Butcher	22	24	41	29	24	22	15	24	27	22	19	18	24	4
Chef	24	43	68	28	49	52	41	51	29	34	44	40	43	5
Composer	23	45	73	28	48	56	42	62	28	33	53	42	42	5
Criminal	21	38	55	23	42	43	42	49	28	29	38	37	37	5
General	24	50	90	28	49	56	41	98	29	53	66	63	43	5
Hacker	22	29	49	25	32	27	24	31	30	17	28	25	29	4
Magician	24	50	89	28	49	55	41	90	29	90	71	73	43	5
Politician	23	50	89	27	48	55	42	87	28	53	72	67	42	5
Priest	24	50	89	28	49	55	41	90	29	56	71	73	43	5
Scientist	24	40	52	25	41	41	35	37	29	28	40	32	43	5
Sculptor	2	3	13	2	2	3	7	1	3	4	3	4	3	5

Table 5. 5 – *Number of Mappings Generated Between Professions Domains*

We can also identify a range effect between the distance in structure-space and the size of the resulting mapping - as shown on Figure 5.5. This indicates that larger distances in structure space produce smaller inter-domain mappings.

The correlation between structural distance and mapping size was seen as quite encouraging, although the correlation was weaker than we would have liked. These results indicate the influence of structure on analogy retrieval, and how this influence extends to the size of the inter-domain mapping that can be formed. However, finding large mappings is not sufficient for a creative analogy, so we will now look at the results of inference generation.

Validation Correlation

Finally, we examined the correlation between the distances in structure space and the number of inferences that were generated. Inferences are divided into two categories. The *immediate* inferences are those that were generated directly

by the CWSG algorithm, and are further divided into the *valid* and the *invalid* predicates. The invalid inferences were passed to *adaptation* and those that were amenable to Kilaza's adaptation process are referred to as the adapted predicates. The total number of inferences then is the immediate inferences plus the adapted inferences.

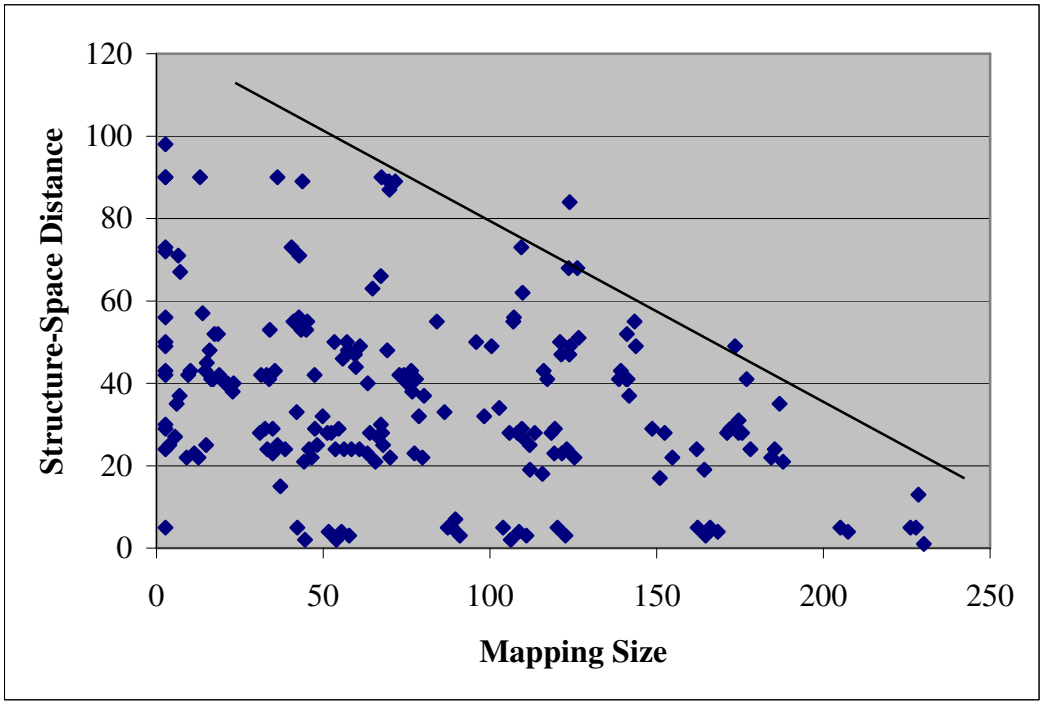

Figure 5. 5 - *Correlation between Structural Distance and Mapping Size for the Professions KB*

We calculated the correlation between the structure-space distance and the number of immediate inferences that were generated using the Pearsons product-moment correlation, and it was found to be low and not reliable r (196) = –0.047, p < 0.513. The correlation between the structure-space distance and the number of adaptations was also found to be low and not reliable r (196) = –0.028, p < 0.697. Finally, the correlation between the structure-space distance and the total number of inferences was calculated and was found to be low and not reliable r (196) = -0.051, p < 0.475. These results indicate that structure

space was ineffective in identifying source domains that provide inferences to the target domain. Of course there is no reason to expect such a relationship, because the number of inferences can vary independently of the number of mappings in the analogy. We also calculated the correlation between the size of the inter-domain mapping and the total number of inferences that resulted using the Pearsons product-moment correlation and was found to be low and reliable $r(196)= 0.167$, $p<0.020$. Therefore, the goal of making a creativity engine did not appear to be supported by our retrieval technique, on this collection of domain descriptions.

In summary, the Professions results indicate a reliable but weak negative correlation between the structure-space distance and the number of mappings that were identified, r (196) = -0.331, p < 0.0005. Thus, the structure of a domain as identified by the structural attributes, seems to play some role in identifying candidate sources that form larger inter-domain mappings.

5.4.2 Correlations in the Assorted KB

As stated earlier, the Assorted KB differ from the Professions KB as its domains are smaller than those of the Professions KB, use a greater diversity of relational predicates, and have a distinct set of high-order predicates. We now examine results from these domains, to see if they confirm or contradict the Professions results.

Retrieval Distances

This collection contains 81 domains generating 6561 analogies, including some of the most frequently referenced analogies like the *solar-system:atom, tumour:fortress* and the *heat-flow:water-flow* analogies. The distances between all domain-pairs were recorded, and a small section of these results are listed in Table 5.5 (the full set of results is contained in Appendix B-1). These inter-domain distances range from 1.41 to 35.63 (M=9.26, SD=6.62).

	3-bears	Apple	Army	Arthurian-saga	Assasinate-JFK	Assasinate-pig	Atom	Atom-clone	atom-falkenhainer	Banker	Beautiful-game	Bird	Burn-paper
3-Bears	2.6	5.7	17.7	7.8	3.5	4.6	3.0	3.0	5.3	2.8	5.8	6.6	5.9
Apple	4.6	2.6	14.4	6.9	5.8	8.1	5.6	5.6	3.5	5.3	6.6	5.7	6.9
Army	14.1	10.3	2.6	11.0	15.5	18.3	15.7	15.7	11.2	15.5	13.8	11.1	14.0
Arthurian-Saga	4.1	3.9	13.7	2.6	4.2	6.6	4.4	4.4	4.5	4.0	2.4	2.0	3.0
Assasinate-Jfk	4.9	7.6	19.1	8.6	2.6	3.5	3.2	3.2	7.1	3.3	5.9	7.4	5.8
Assasinate-Pig	6.7	10.0	21.7	10.5	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4
Atom	4.6	7.4	19.2	8.7	3.2	3.0	2.6	2.6	6.9	2.8	6.0	7.5	5.9
Atom-Clone	4.6	7.4	19.2	8.7	3.2	3.0	2.6	2.6	6.9	2.8	6.0	7.5	5.9
Atom-Falkenhainer	4.0	3.5	15.0	7.2	5.2	7.5	4.9	4.9	2.6	5.0	6.6	5.7	6.5
Banker	4.5	7.2	19.1	8.5	3.3	3.2	2.8	2.8	7.0	2.6	5.9	7.4	6.0
Beautiful-Game	3.7	6.0	16.8	5.5	1.7	3.5	2.0	2.0	5.9	1.7	2.6	4.6	2.8
Bird	3.5	3.5	14.2	4.0	3.3	6.0	3.5	3.5	3.6	3.3	3.0	2.6	3.2
Burn-Paper	3.9	6.2	17.0	5.7	1.4	3.3	1.7	1.7	5.8	2.0	2.8	4.7	2.6

Table 5. 6 – *A Small Sample of the 6561 Structure-Space Distances from the Assorted Domains*

Mapping Correlation

The inter-domain mappings that resulted from these analogies were recorded, and ranged in size from 1 to 15 predicates (M=1.57, SD=1.14).

We calculated the correlation between the structure-space distance and the size of the inter-domain mapping using Pearsons product-moment correlation and found it to be low and reliable r (6561) = -0.074, $p < 0.0005$. This correlation value indicates that structural attributes had no influence on identifying promising candidate sources from the Assorted KB. Therefore, for this KB structure-based retrieval does not appear to be usefully supported by structural features.

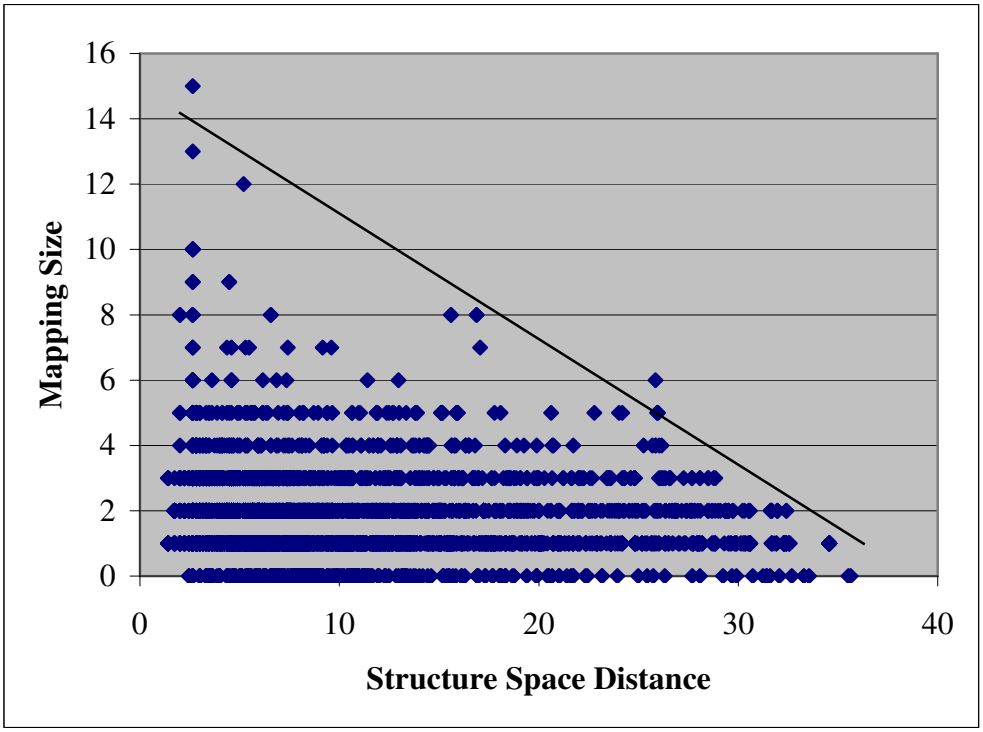

Figure 5. 6 - *Scatter-plot of the Relation Between Structural Distance and the Size of the Inter-domain Mapping*

However, a range effect was identified between structure space and the size of the resulting mapping – similar to that observed on the Professions KB. This again indicates that larger distances in structure space tend to produce smaller inter-domain mappings – as indicated on Figure 5.6. Thus, while individual correlation values are low, there is a broader tendency for structure-based retrieval to identify potentially larger mappings.

However, the correlation results between structure space and mapping size differs on the two collections of domains. We will explore possible reasons for this discrepancy in Section 5.4.3, but first, we will examine the validation results.

Validation Correlation

The number of inferences that were generated by these analogies was recorded. The number of validated inferences ranged from 0 to 8 (M=0.44, SD=0.83). We calculated the correlation between the structure-space distance and the total number of valid inferences using Pearsons product-moment correlation and found it to be low and reliable r (6561) = -0.056, P<0.0005. The correlation between the structure-space distance and the number of adaptations also was found to be low and reliable r (6561) = -0.043, P<0.0005. Finally, the correlation between the structure-space distance and the total number of inferences was found to be low and reliable r(6561)= -0.063, p<0.0005. Again, retrieval appears ineffective in identifying candidate sources that can supply inferences to an arbitrary target domain. We also calculated the correlation between the size of the inter-domain mapping and the total number of inferences that were generated and found it to be low and reliable r(6561)= -0.195, p<0.0005. All these results indicate that structure based retrieval had little ability to identify useful domains.

We also calculated the correlation between the distance in structure-space and the number of high-order inferences and was found to be low and reliable r(6561)= -0.082, p<0.0005. We then calculated the correlation between the size of the mapping and the number of high-order inferences and was found to be low and reliable r(6561)= 0.130, p<0.0005. Finally we calculated the correlation between the distance in structure-space and the total number of inferences (first-order and high-order) and was found to be low and reliable r(6561)= -0.073, p<0.0005.

The overall performance of structure-based retrieval on the Assorted KB showed that structure based retrieval was ineffective in this collection of domains. These domains included some of the most referenced domains in the analogy literature (including the *solar-system* and *atom* domains) and the other domains were comparable in size. These results indicate that structure-based analogy retrieval does not appear to be good at identifying sources that form large mappings in this Assorted KB.

5.4.3 Comparison between Professions and Assorted Results

Some of the results obtained using the Professions KB are at odds with the results from the Assorted KB. The retrieval-to-mapping correlation figure for the Professions database was r (196) = -0.331, $p < 0.0005$, which is notably stronger than the corresponding figure obtained from the Assorted domains collection at r (6561) = -0.080, $p < 0.0005$. The first correlation figure indicates that the inter-domain distance in structure space has *some* influence on the size of the mapping that results, while the latter figure suggests that it does not. The explanation for this discrepancy would appear to lie in the difference between the two KBs.

Relation	**Number of Instances in the Professions knowledge base**
part-of	313
depend	106
control	85
affect	77
create	55
perform	49
location-of	25
effect	22
substance	19
wear	11
disconnect	5
mode	5
purpose	2
down	1
up	1
is-a-minus	1

Table 5. 7 - *Relational Predicate usage in the Professions Knowledge base*

There are two main differences between the Professions and Assorted KBs that might account for these differing results. First, there is a significant difference in domain sizes in the two collections. Each of the Professions domains range from 10 to 105 predicates while each of the Assorted KB items range from 1 to 15 predicates. Thus, the 14 Professions domains are quite spread

out across the structure-space. In contrast, the 81 Assorted domains are more densely packed into the lower values of the attributes of structure space. Thus, it may be that Kilaza operates more effectively when structure-space is less densely populated. We will investigate this possibility in detail in section 5.5.

The second major difference between the Professions and Assorted KBs is the diversity of relational predicates used in the two collections. Each of the 14 Professions domains use multiple instances of a small set of relations, including: `affect`, `control`, `create`, `depend`, `part-of` and `perform` (see Table 5.7 for the complete list). The Assorted domains use a greater diversity of relations, and multiple instances of relational predicates are very rare (see Table 5.8 for a full list of the Professions relations).

The average number of duplicate predicates in a domain in the Professions KB is M=46.5 (SD=28.1). In contrast, the average number of duplicate predicates in the 81 Assorted domains is M=0.678 (SD=1.183), meaning that almost all predicates are unique within each domain. We can also think of these numbers in terms of the ratio of duplicate predicates to distinct predicates within each domain, and the average ratio of duplicate predicates in the Professions domains is M=0.80 (SD=0.13). The average ratio of duplicate predicates in the Assorted domains is M=0.119 (SD=0.20). The increased relational diversity has a direct impact on the ease with which an inter-domain mapping may be generated (see Figure 5.7).

When both source and target domains use only one relation each, then the structure of the relations plays a primary role in determining the size of the largest inter-domain mapping that can be generated. This is only limited by the 1-to-1 constraint, which ensures that no object is mapped twice.

As we increase the number of relations in each domain, this tends to reduce the size of the largest mapping that can be found. This is because the 1-to-1 constraint now also applies to the relations of the mapping. The more distinct relations that are used, the less likely it is that two structurally similar domains will form a large inter-domain mapping.

Relation	Count	Relation	Count	Relation	Count
Part-of	37	Become	2	Flow-along	1
Has-part	20	Born-in	2	Found	1
Cause	19	Bounce	2	Go-to	1
Connect	16	Contains	2	Has	1
Enable	16	Converge	2	Hear	1
Control	15	Decorate	2	Hop	1
Made-of	10	Died	2	Influence	1
Inside	8	Drive	2	Inhabits	1
Attracts	7	Expose	2	Injure	1
Heavier	6	Facilitate	2	Jealous-of	1
Loves	6	Help	2	Keep-out	1
Attack	5	Hit	2	Kick	1
And	4	Hoards	2	Lead-to	1
Create	4	Lifestyle	2	Live-in	1
Go-down	4	Lived-in	2	Lives-in	1
Greater	4	Obtain	2	Located-in	1
Holds	4	Opposite-sign	2	Lusts-after	1
Next-to	4	Own	2	Make	1
Propel	4	Paddle	2	Melt	1
Style	4	Plays	2	On-top-of	1
Affect	3	See	2	Owns	1
Avoid	3	Temperature-of	2	Played-with	1
Cut	3	Treatment-of	2	Product	1
Damage	3	Type-of	2	Proportional	1
Directed-line	3	Use	2	Repair	1
Eat	3	Used-for	2	Result-in	1
Find	3	Build	1	Shoots-with	1
Flow-from	3	Burn	1	Sit-in	1
Flow-to	3	Buy	1	Sit-on	1
Line	3	Capture	1	Subject-of	1
Revolves	3	Conquer	1	Support	1
Split-into	3	Cover-state	1	Tell	1
Surround	3	Enter	1	Theme	1
Taller-than	3	Examine	1	Throw	1
Transport	3	Execute	1	Thud	1
Assassinate	2	Flies-through	1	Works-in	1

Table 5. 8– *The Assorted Domains Use a Variety of Relations*

Therefore, we are left with two alternate explanations for the discrepancy in the retrieval-to-mapping correlation figures. Both the "badly indexed domains" (or structural uniqueness) argument and the "relation diversity" arguments seem plausible, but we wish to identify which of these explanations is correct. In the next section, we describe an additional set of tests that allow us to select which of the two explanations is correct.

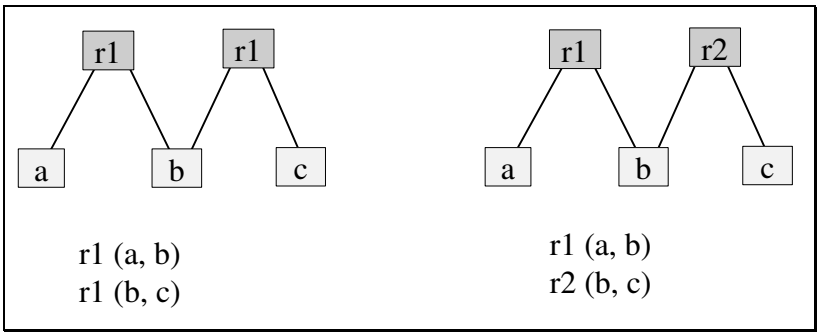

Figure 5. 7 – *Increasing the Variety of Relations in a Domain, Reduces the Chances of Isomorphic Domain-pair Forming a Large Inter-domain Mapping*

5.5 Structure Space and Graph-Structure

The first of these two explanations basically states that structure space provides poor descriptions of the graph-structure of small domains – and that this caused the poor performance on the Assorted KB. If the graph-structure is poorly described, we might expect to see a large amount of clustering in structure space, as many different graph-structures are bundled into the same location in structure space. This strikes at the heart of our technique for supporting structure-based retrieval and thus we shall explore this hypothesis in some detail.

All of the Professions domains were uniquely indexed in structure space, that is, no two domains had the same structural index in structure space. The graph-structure of 45 of the 81 Assorted domains were uniquely described by their structural index. There were 6 pairs of co-located domains, 2 triples and

one group each containing 4, 6 and 8 domains. This indicates a relatively small amount of clustering of domains in structure space, but structure-space generally provides adequate descriptions the structure of these domains.

5.5.1 The Alpha-Numeric Domains

The results of tests using the Professions and Assorted KBs could be explained by the possibility that structure space itself does not provide adequate support for structure-based retrieval. In this section we explicitly test the ability of structure space to identify structurally similar (isomorphic and homomorphic) domains, from a collection of structure rich domain descriptions. This test will help us determine if structure-based retrieval can be expected to perform any better on small domains which were not created as part of this project.

To this end, we created a corpus of 62 structure-rich domains that are described by just one relation. These domains are symbolic representations of the alpha-numeric characters represented on a 16-segment display, as illustrated in Figure 5.8. This corpus uses only one semantic relation (`line`), and the "objects" are vertices where line-segments meet on the 16-segment display. This collection enabled us to determine the *precision* and *recall* of the structure-based retrieval model. The structure of all domains in this corpus are depicted in Figure 5.9.

This collection contains many alphanumeric characters that are isomorphic with one another. For example the characters *N, Z, U,* and *C* are all composed of a sequence of 6 line-segments, and so they are all co-located at the same point in structure space (Figure 5.10). Similarly isomorphic groupings include *O* and *0,* and *n* and *u*. All isomorphic domains were included in the data set, so when one member of an isomorphic collection is presented for retrieval, all isomorphic domains will also be returned. Of course, this will necessarily affect the *precision* of the retrieval process, but isomorphs must be allowed as they also occur in real analogy retrieval problems.

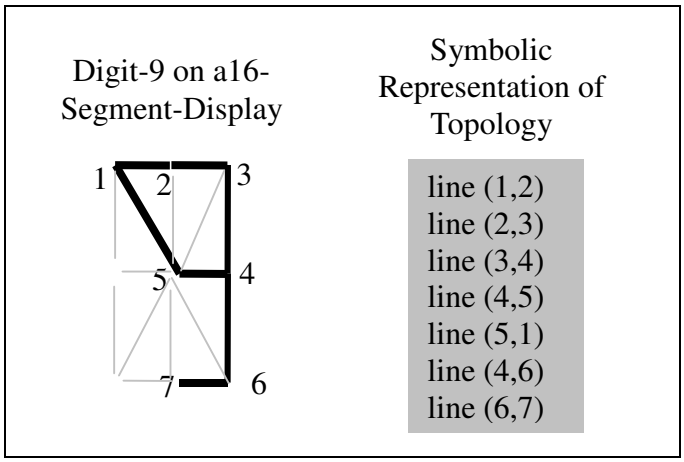

Figure 5. 8 - *Symbolic Representation Derived from a 16-Segment Display*

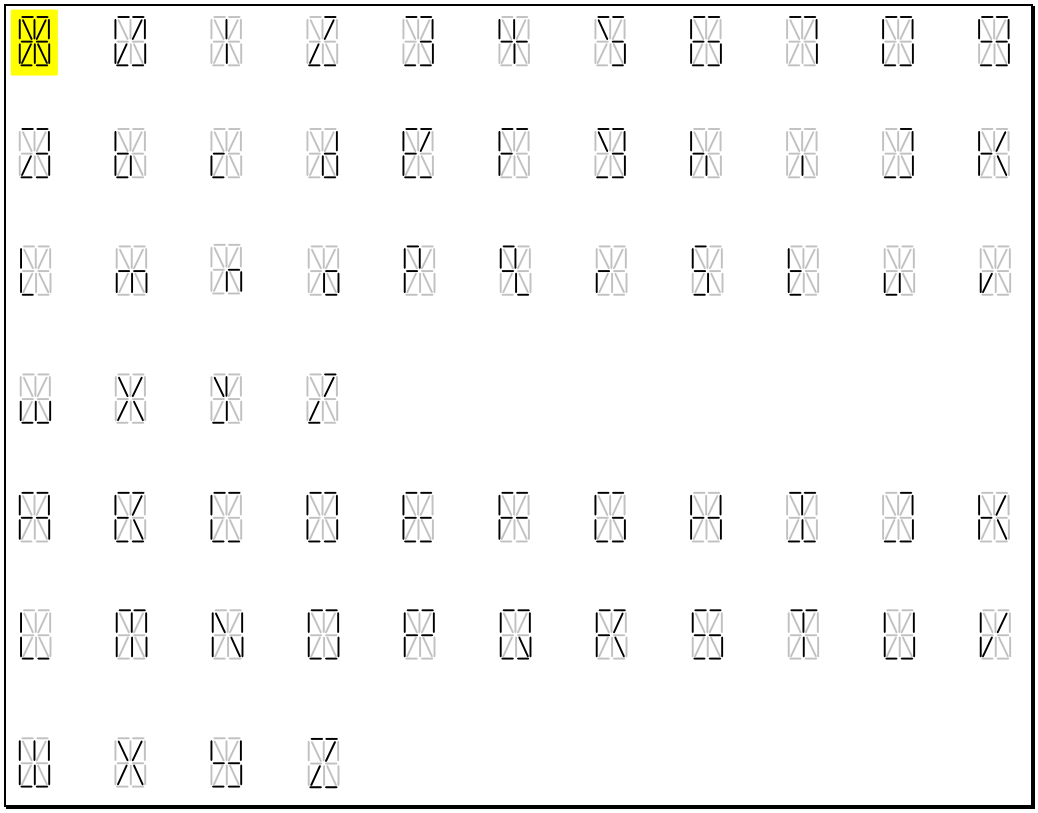

Figure 5. 9 - *The16-Segment Display Test-suite*

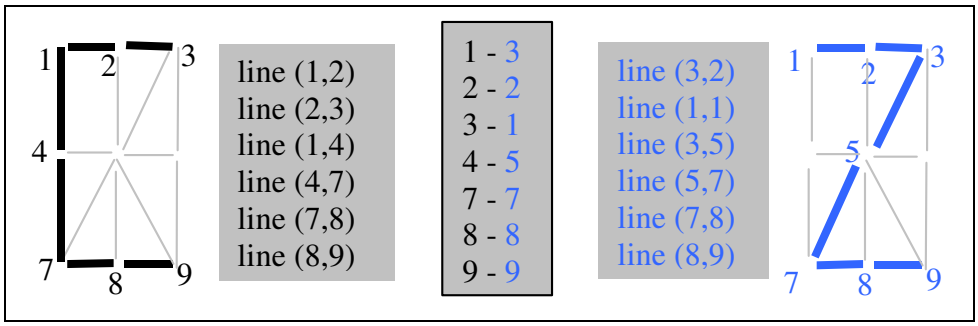

Figure 5. 10 –*Isomorphism Between Two Alphanumeric Domains, Detailing the Inter-domain Mapping*

We use two standard metrics to measure the accuracy of the retrieval process; *precision* and *recall* (Jurafsky and Martin, 2000).

```
Precision = Rel-Ret / Rel

Recall = Rel-Ret / Ret
```

Where *Rel* is the number of relevant items in a collection, *Ret* is the number of items retrieved from that collection, and *Rel-Ret* is the number of relevant items retrieved from the collection.

5.5.2 Precision and Recall on the Alpha-Numeric Domains

This set of tests followed the same procedure as the earlier experiments. Each domain contained in memory was taken in turn and was used as the target. All domains located at the same point as the target within structure space were retrieved. (We did not displace the locus-of-retrieval in this experiment, as this would make our results more difficult to interpret).

In each of the 62 retrieval episodes, all domains at the same location as the target domain were retrieved from structure space. In each case, the target domain was among those identified from memory, giving a *retrieval* figure of 100%. This result was expected as all structurally identical domains must necessarily have the same index in structure space, and will therefore be retrieved.

We also examined how accurately structure-space represented the domain's structure, by examining how many other domains were among those retrieved - the *precision* of structure-based retrieval is illustrated in Figure 5.11. Just under 10% of the domains are uniquely indexed by the structural attributes. Approximately another 10% of the domains were co-located with just one other domain in structure space. The average precision value across all trials was 0.379 (SD=0.22), and the mode of the number of domains retrieved was 3.

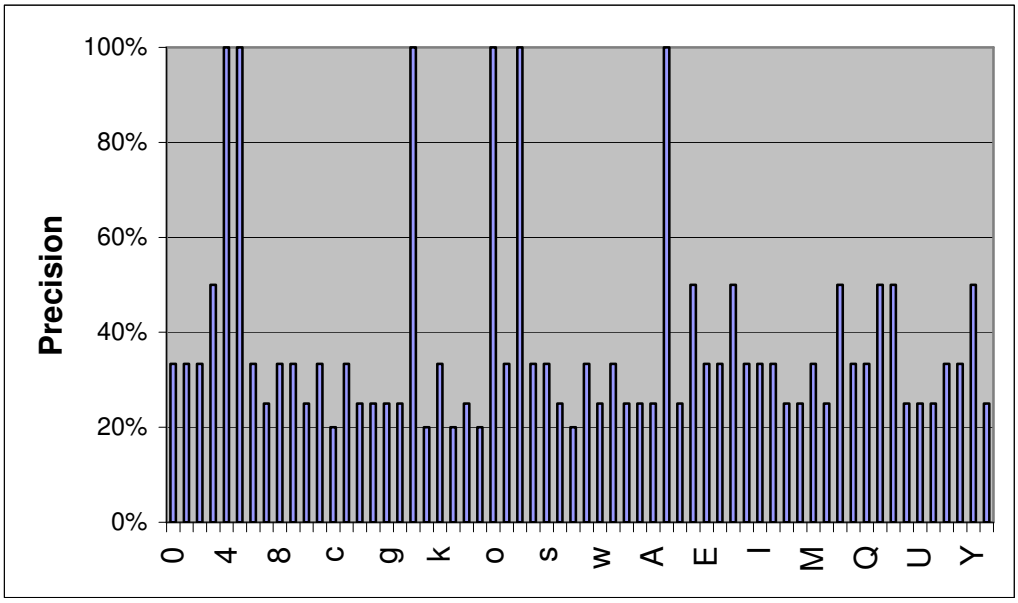

Figure 5. 11 - *Kilaza's Recall on Alphanumeric Domains*

While these precision results appear quite low, we point out that there is a large amount of structural duplication among these structures. For example, the lower case letters *u, n, j, l,* and *c* form one isomorphic group that essentially consist of a sequence of three line segments –differing only in location and orientations on the 16-segment display. The structural attributes described above do not differentiate between these, thereby generating the group whose members are retrieved with 20% precision. We point out that all members of this group can be mapped onto one another (akin to the example in Figure 5.10). Thus, we

conclude that the structure retrieved are in fact analogous to one another – and that structure-based retrieval does indeed support analogy retrieval.

The retrieval experiments conducted on the alphanumeric domain indicate that structure-based retrieval performs adequately on this collection of domains. The *recall* of 100% was expected and mimicked the results produced on the other two collections of domains. The *precision* results were quite positive, given the prevalence of isomorphic structures within this collection.

5.5.3 Implications for Structure-based Retrieval

We have now conducted three separate sets of experiments to determine the usefulness of structure-based retrieval, using the Professions, Assorted and Alpha-numeric collections. The features of structure space appear to represent domain structure adequately and provide support for structure-based retrieval even on small domains. Thus the discrepancy between the two retrieval performance values noted earlier, does not appear to be a result of the inability of structure space to adequately represent the structure of the small domains in the Assorted KB. This suggests that the discrepancy must be a result of the greater semantic diversity occurring in the Assorted KB, making it more difficult to form larger inter-domain mappings.

Our conclusion must therefore be that structure-based retrieval operates best when structure dominates over semantic factors, in determining how two domains form an inter-domain mapping. As the influence of structure decreases and is replaced by greater semantic diversity, the usefulness of structure in identifying candidate sources decreases rapidly. Structure-based retrieval then appears to be a useful technique in identifying candidate sources, when these are described by a smaller set of (generalised) relational predicates.

5.6 Creativity Tests

In this final set of experiments, we examine the model's ability to discover the appropriate source domain for 10 target problems. These experiments involve presenting the target domain of some famous analogies that are widely regarded

as being creative, to see if Kilaza identified the (historically) correct source. The test analogies included Rutherford's *solar-system:atom* analogy, the *general:surgeon* and the *heat-flow:water-flow* analogy (Boden, 1994). Each of these analogies is based on identifying a specific source domain for each given target. Kilaza performs these retrieval experiments without using the relation names used in the target domain description. The objective of this experiment is to assess how many of the correct (creative) sources are retrieved by Kilaza.

The results presented so far treated all domains as candidate sources, as every possible analogy was generated. In this final set of experiments, we apply a hard constraint within structure space to identify a subset of the possible sources to serve as candidate sources. Only these candidate sources will be used to generate analogies with the given targets.

This experiment uses a memory containing 158 domain descriptions, containing the Professions, Assorted and Alpha-numeric KBs. We identify as candidate sources, all domains located within a distance of 10 units from the target's locus of retrieval. This metric ensures we select all domains that are structurally similar to the target, and ignore those that are considered structurally different. We point out that this is not a relative metric, used to rank domains according to similarity (though it could be used for this purpose). It generates an absolute measure of structural similarity, and thus may select a different number of candidate sources for each problem.

A series of 10 tests were performed on the retrieval model. These tests involved presenting each target problem (Column 1 of Table 5.9) to the model in turn and recording all candidate sources returned. The results of these experiments are summarised in Table 5.9. All of the desired creative sources (Column 2 of Table 5.9) were among the identified candidate sources, giving a *recall* value of 100%. This means that all creative sources were among those selected from memory, being located within a distance of 10 units from the targets locus of retrieval. This was an important result, indicating the usefulness

of structure-based retrieval for identifying creative analogies. Even more important is that all the analogies were identified from the Assorted domains.

Figure 5.12 illustrates the *precision* of the retrieval algorithm for these 10 problems. The precision values for these creativity tests were quite low, but this was expected because of the unpredictable nature of creativity. Interestingly the precision values for two problems were quite high, indicating that the desired source was among the few candidate sources that were retrieved. These precision values indicate that structure-based retrieval is a very useful tool in identifying creative analogies. Furthermore, Kilaza's retrieval process will be equally successful if the two domains are described using a different relational terminology.

Target	**Required Source**	**Retrieved**	**Structure-Space Distance**
Atom	Solar-System	y	6.082763
Atom-Falkemhainer	Solar-System-Falkenhainer	y	3.464101
General	Surgeon	y	6.557439
Heat-flow	Water-Flow	y	3.316625
Leadbelly	Caravaggio	y	4.3588989
Love-triangle	Triangle-Directed	y	4.795832
Requited-love	Love Triangle	y	7.937253
Bird	Fish	y	7.416198
Banker	Vampire	y	3.316625
Cycling	Driving	y	*3.162278*

Table 5. 9 – *Retrieving the Creative Source Domains*

In the next chapter, we will examine the inferences from each of these analogies to see if the correct inferences were also generated and successfully validated.

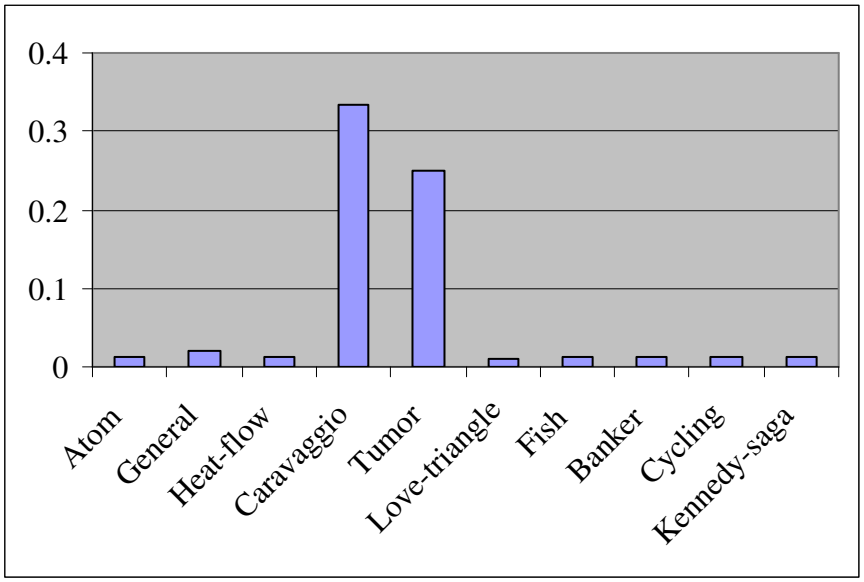

Figure 5. 12 – *Retrieval Precision for the 10 Creative Analogies*

5.7 Structure, Representation and Rerepresentation

The previous section described how Kilaza identifies well known creative analogies - based on the standard representation of these domains that is found in the literature. However, many of these descriptions are based on a *post hoc* description of the domains involved, and are heavily influenced by the analogies themselves. These *post hoc* descriptions explicitly highlight the relational commonality between the source and target domains. Thus, they do not accurately describe the domains as they would have existed before the analogies were first created. Consequently, modelling creativity using these representations makes the creative analogy problem *significantly* simpler.

For example, consider the representation of the *solar-system* and *atom* domains (O'Donoghue and Crean, 2002), described in Figure 5.13. We argue that these representations are a more accurate depiction of the domains, as they were conceived *before* Rutherford's h-creative (Boden, 1992) analogy was

discovered. A crucial feature of this representation, is that it highlights the difference between the "fundamental forces" that operate in the two domains. The distinction between the four "fundamental forces" is a core distinction that applies right across science. (The four fundamental forces are `gravitation`, `electromagnetc-attraction`, `weak-nuclear-force` and the `strong-nuclear-force`). Ernest Rutherford would most likely have thought of the target relation between the `nucleus` and `electron` as `electromagnetic-attraction`, and not the more generic `attracts` relation. The corresponding relationship between source's `sun` and `planet` is `gravitation`. It is only after he found the analogy, which involved mapping the `electromagnetic-attraction` with `gravitation`, that these relationships can be generalised to their common super-class, like `attracts` (Gentner, 1983).

Differences in domain terminology is one of the crucial differences between elaborating a given analogy, and the much more difficult task of generating a novel *h-creative* (or *p-creative*) analogy (Boden, 1992). These differences are particularly prevalent when the first-order relationships describing the problem domains originate in different disciplines. When modelling analogical creativity, we must expect to encounter these differences in terminology, and our models of retrieval and mapping must be able to overcome these problems.

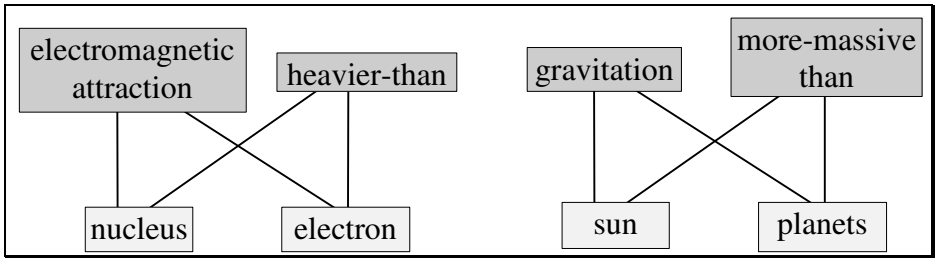

Figure 5. 13 –*Relations in the S*olar-system *and* Atom *domains before the analogy was created*

Kilaza's retrieval model is oblivious to these variations in domain terminology. This is because terminological details do not impact upon the domain's structure, and thus have no influence on retrieval. So, Kilaza will perform retrieval as well on the "standard" description of these domains (Figure 5.3) as on the more realistic descriptions provided above (Figure 5.13). Furthermore, while the mapping model of Kilaza has a preference for identical relations, it can generate mappings in the absence of predicate identicality. Therefore, we conclude that Kilaza is better suited to the task of generating creative analogies, than other analogy models.

The ARCS (Thagard et al, 1990) model combines structure with semantic influences when performing analogy retrieval. It could therefore retrieve and map creative analogies like that of Figure 5.13. Of course, the semantic difference between the two "attracts" relations will reduce the goodness of the analogy that is found. The most limiting problem with the ARCS model is its difficulty in generating inferences, which are central to any useful analogy.

5.7.1 Problem Rerepresentation

The retrieval model in MAC/FAC (Law *et al*, 1994) uses predicate identicality to identify candidate sources. Therefore, it is not capable of generating the more creative version of the *solar-system:atom* analogy (Figure 5.13). Furthermore, the current versions of SME (Falkenhainer *et al*, 1989; Forbus, Oblinger, 1990; Forbus, Ferguson, Gentner, 1994) employs predicate identicality as a hard constraint, and so can not find mappings between non-identical predicates.

Gentner *et al* propose a new approach to mapping non-identical relations called *rerepresentation* (Yan *et al*, 2003). This is a process that is distinct from analogy itself, but can interact with it. However, they have not yet described how *rerepresentation* might be applied to analogy retrieval, making an accurate assessment difficult.

However, the *rerepresentation* process re-construes parts of the two domains in an analogy to improve the match. While they propose a number of solution strategies, we will only discuss those that are relevant to our current discussion. Rerepresentation uses two methods to overcome the identicality constraint (Gentner, 1983), by applying *transformation* rules which can re-write predicates to improve the inter-domain mapping. Thus *transformation* might replace the `electromagnetic-attraction` relation with `sub-atomic-attraction` or `attracts`, so that the necessary mapping can be identified. SME will therefore become capable of evaluating the more challenging version of the *solar-system:atom* analogy, if this rerepresentation process is incorporated with SME. However, this still does not address how the retrieval component of MAC/FAC might retrieve these non-identically represented problems.

5.8 Conclusion

Finding a novel analogy involves finding new semantically distant candidate sources, which can form a mapping and supply inferences to some given target. Instead of using semantics as a basis for retrieval, the Kilaza model investigated the use of domain structure for selecting candidate sources. This approach offered the possibility of overcoming the usual semantic constraints associated with the existing approaches to analogy retrieval.

We examined the performance of structure-based analogy retrieval on two different collections of analogy domains - the Professions and Assorted KBs. Kilaza's structure-based retrieval model was not appreciably successful on either collection, in identifying domains that provided large numbers of inferences to a given target domain. We also tested the retrieval model on finding sources that generate large inter-domain mappings with a given target, and these results were much more positive. Retrieval tests using the Professions domains showed a negative correlation of -0.331 between the distance in structure space and the size of the mapping. This highlighted that structurally similar sources tended to form later inter-domain mappings, but testing on the Assorted KB found no correlation.

Resolving these conflicting findings necessitated a further set of tests, using the alpha-numeric domains. These tests revealed some constraints on the applicability of structure-based retrieval, showing that its operates best when domains are described by a relatively small number of relational predicates.

We also tested Kilaza's ability to identify the well known *h-creative* sources for a number of famous problems, including the *solar-system:atom* and the *tumour:fortess* analogies. The Kilaza model correctly identified all required sources, by using a threshold distance in structure space to identify candidate sources. Successful retrieval on these sample problems indicates that Kilaza's retrieval model does provide a useful model for discovering novel and creative analogies.

Chapter 6

Validation Results

"One good analogy is worth three hours discussion."

- - Anon.

"Analogies decide nothing that is true, but they make one feel more at home"

- - S. Freud, New Introductory Lectures on Psychoanalysis, 1935.

6.1 Introduction

Validation is the third phase in Kilaza's three-phase model for finding novel analogies. The validation model is necessary to detect any false analogies that we might inadvertently expect to encounter, when searching for creative analogies. Validation will examine the inferences generated by all analogies, rejecting those it considers to be invalid. Broadly speaking, we describe invalid inferences as those consisting of an unacceptable combination of relation and arguments, such as `sleep(ideas, furiously)`.

Analogical validation may not be of great importance when the two domains are carefully selected and described to eliminate the possibility of

unwanted inferences - such as when analogies are used in education or argumentation. However, when searching for novel and creative analogies, validation assumes a more crucial role. It is validation that must detect and reject the numerous invalid analogies and inferences that will inevitably be encountered on the road to finding a creative source domain. So finding creative analogies involves not only finding novel sources, but also requires detecting and rejecting many unwanted analogies and their inferences.

The previous chapter presented some broad results on the quantity of inferences that resulted from a selection of analogies. In this chapter, we will look at the details of these and many other inferences, focusing on how the validation process dealt with them. The objective of this chapter is to assess how well the validation process performed when presented with the inferences that were generated from analogies retrieved from the Professions and Assorted KB.

6.1.1 Structure of this Chapter

In this chapter we assess the performance of Kilaza's validation model. The chapter is composed of two main sections. The first section analyses the performance of the validation model on the analogies retrieved from the Professions KB, while the second section deals with the Assorted KB's retrievals. Each section is in turn composed of three main sub-sections. First, we present an overview of the validation results generated by Kilaza, quantifying the number of inferences that were classified as *valid, invalid* and *adapted*. The second part details experiments that were conducted to determine how well Kilaza performed the *validation* process. The last part details the results of experiments to determine how well the *adaptation* process performed. The chapter concludes with some general observations on the validation process.

6.2 Experimental Set-up

The Profession and Assorted KBs are again used in this chapter, but now serve to test the performance of Kilaza's *validation* model. The same experimental set-up is used to analyse Kilaza's performance on the two

different knowledge bases. But first we look at the experimental set-up and the categories of inference that Kilaza uses.

A memory was created containing all domains from the relevant knowledge base, all of which served as candidate sources. Next, each of these domains were taken in turn to serve as the target problem. The candidate sources were retrieved in turn (based on their relative distance in structure space) and the inter-domain mappings were identified. All inferences mandated by each of these analogies were generated and passed to the validation process. As described in Chapter 4, Kilaza categorises all inferences as either *valid* or *invalid*. *Valid* inferences being those inferences that contain no identifiable deficiencies and *invalid* inferences are those which are rejected by validation. The results of all validation activities were recorded for further analysis. Finally, all *invalid* inferences were passed to Kilaza's *adaptation* process and again, all successful adaptations were recorded for further analysis.

We refer to the combination of *valid* and *invalid* inferences as *generated inferences,* because these are produced directly by the CWSG inference. In contrast, the *adapted* inferences are generated by the validation model, and thus are treated quite separately. In the following results, we make a clear distinction between generated and adapted inferences.

All inferences created by Kilaza therefore fall into one of these three categories: *valid, invalid* and *adapted* predicates. The main focus of this chapter is to investigate if these categorisations are in fact accurate. In this instance, accuracy was assessed by comparing Kilaza's categorisations with those sanctioned by people. Experiments were performed to obtain goodness ratings for the *valid, invalid* and *adapted* inferences. This involved two human raters who rated the quality of inferences in each of the three categories. In Experiment 1 we used the inferences generated by the Professions domains, and in Experiment 2 we used the Assorted domains.

The raters were asked to give each predicate a rating between 1 and 7. A rating of 1 represented a predicate that could not be considered credible under any circumstance, while a rating of 7 represented a predicate that could

certainly be considered credible in some circumstance. A rating of 4 represented a predicate that was not obviously either credible or not credible in any circumstance.

These ratings were used to determine the 'correct' class for each predicate, from which we determine Kilaza's accuracy. Predicates were considered to be rated as *valid* by the human raters under two circumstances. Firstly, when both raters awarded a score of more than 4 to a predicate. Secondly when a predicate was rated as valid by just one rater, it was deemed to be given an ambiguous rating and was considered *potentially* true. (Because our original objective was to reject clearly *invalid* predicates, it was considered necessary to treat these ambiguous predicates as valid). Treating potentially valid predicates as valid was also appealing from a creativity point of view, as it did not reject the more creative predicates that may be generated. A predicate was therefore considered to be rated as *invalid* only if both raters awarded a score of less than 4.

Finally, we considered that the two raters disagreed about the rating of a predicate if one rater awarded it a score of more than 4, while the other awarded it a score of less than 4.

It should be pointed out that assessing the creativity of Kilaza could not rely on the standard methodology employed by cognitive science (Eysneck and Keane, 1995). Assessing the performance of Kilaza required the use of a novel methodology. Traditional cognitive science assesses peoples performance at some task, and generates models based on these observations. The goodness of the model is assessed by how closely the model fits the prior observation. However, this project was based on a different methodology. The computational model was used to make predictions about how people would assess the inferences generated by the creativity model - and these predictions were then assessed by people. The output generated by the Kilaza model was assessed by having raters assess the goodness of the contents of each of its output categories. The details of the assessment process will be made clear during this chapter.

6.3 Overview of the Professions Inferences

In the first half of the chapter, we analyse validation's performance on the inferences generated by the Professions domains. The 14 Professions domains generated 196 different analogies. The inferences generated by these analogies form the materials used in the experiment. The Professions domains contained no high-order relations, thus Kilaza could not generate any high-order inferences (and so these inferences did not pass automatically through validation). So, all results generated from the Professions domains relate to first-order inferences.

Kilaza's model of mapping and inference generated 175 inferences from the Professions analogies, and these inferences were passed to its validation phase. Kilaza's validation model classified 151 (86.2%) of these inferences as *valid*, while 24 (13.7%) inferences were classified as *invalid* (see Table 6.1 below). So in effect, the validation model is detecting invalid inferences from the total set of inferences generated by Kilaza.

Type of Inference	*Number of Inferences*
Valid Inferences	151 (86.2%)
Invalid Inferences	24 (13.7%)

Table 6. 1 - *Summary of Validation on the Professions Inferences*

An additional 20 predicates were generated by Kilaza's adaptation process - which was invoked on the inferences that were classified as invalid. Thus, 20 of the 24 (83.3%) *invalid* inferences were amenable to Kilaza's adaptation process, and just 4 (20%) *invalid* inferences could not be adapted by the adaptation process.

From the results of the validation process, the first observation we can make is that the Professions domains did in fact generate *invalid* inferences - though only a small number of them. This result supported the need for a validation process - despite the fact that these domains relate to different professions that are all described by a small set of relation types.

Secondly, we note that *all* validations were performed by 'Validation mode 3' - functional attribute validation (Chapter 4). None of the inferences were validated by either 'Identical predicate validation' or 'Partial predicate validation' (validation modes 1 and 2 in Chapter 4). This was unexpected given the large number of predicates stored across the 14 domains held in memory, and the relatively small set of relations that were used to describe these domains. However, none of the generated inferences were sufficiently similar to any of the pre-existing predicates to support these validation mechanisms.

6.4 Experiment 1 - Validation and the Professions KB

The objective of this first experiment was to assess the accuracy of Kilaza's validation and adaptation processes by obtaining human opinions of the classifications given by Kilaza. This consisted of firstly assessing the accuracy of Kilaza at distinguishing between *valid* and *invalid* predicates. This assessment involved examining the predicates that Kilaza placed in each category, and asking people to rate the goodness of the predicates in each category. We then assessed the goodness of the predicates that were generated by Kilaza's *adaptation* process.

The main result we expected to find from this experiment was that the *valid* inferences should receive a higher rating than the *invalid* inferences. This will confirm Kilaza's ability to distinguish between the two categories of inference. Additionally, we expected that the *adapted* inferences would receive ratings that were broadly in line with those of the *valid* category - because the same intra-predicate constraints used for validation were also used for adaptation.

We now describe the experiment that was carried out on the validation and adaptation inferences.

6.4.1 Method

Materials

The materials used were the inferences generated by Kilaza on the 196 Professions analogies. Kilaza classified 151 of these inferences as *valid* from

which 40 were randomly selected for rating. There were no duplicate predicates in this selection. Of the 196 analogies, Kilaza classified 24 predicates as *invalid*, however 3 invalid predicates occurred twice. Therefore these duplicates were removed to give a more accurate picture of the adaptation results. So 21 *distinct invalid* predicates were selected for rating. All 20 *adapted* predicates were selected for rating, as there were no duplicates in these predicates. In summary, 40 valid predicates, 21 rejected predicates and 20 adapted predicates were selected for rating.

Participants and Design

Two raters were used and both raters were familiar with predicate calculus representation.

All data from the three different categories were presented together in a random order. The different categories were not treated as independent variables, therefore the design was a single within-subjects design.

Procedure

The raters were given a spreadsheet containing the list of predicates to be rated. This list was composed of the *valid*, *invalid* and *adapted* inferences, presented in random order. The raters were asked to give each predicate a rating between 1 and 7, where 1 represented a predicate that could not be considered true under any circumstance and 7 represented a predicate that could certainly be considered true in some circumstance.

6.4.2 Results and Discussion

We will discuss the accuracy of Kilaza's *validation* process, before discussion of its accuracy at *adaptation*, as validation occurs before the adaptation process.

Validation Results and Discussion

The results of this Experiment on the Validation process are summarised in Table 6.2

		Human Rating		
	Validation Accuracy	**Rated Valid**	**Rated Invalid**	**Total**
Kilaza Class	**Valid**	17 (42.5%)	23 (57.5%)	40 (100%)
	Invalid	2 (9.5%)	19 (90.1%)	21 (100%)

Table 6. 2 - *Accuracy of Validation of the Professions Inferences*

The average rating awarded to the predicates that Kilaza categorised as *valid* was M=2.62 (SD=2.09), while the average rating awarded to the *invalid* predicates was M=1.57 (SD=1.23).

A McNemar's test was also performed to compare Kilaza's classifications to the categorisations awarded by the raters. The results were: #Invalid-RatedGood = 2, #Valid-RatedBad = 12, p <= 0.0129 showing strong agreement between the two ratings.

Of the 40 predicates classified as *valid* by Kilaza, 17 (42.5%) were rated as valid or potentially valid by the raters. Therefore, 57.5% of the predicates categorised as *valid* by Kilaza were deemed to be invalid by the human raters. This discrepancy between Kilaza and the human raters can be accounted for by two reasons. Firstly, many of the relations used in the Professions domains were not defined by the functional attributes that are necessary for validation, and so Kilaza could not be expected to validate these inferences. Secondly, many of the Professions objects are described by very few attributes. These two factors inflated the number of invalid predicates that were inadvertently accepted by Kilazas' validation process. Improving this performance can be achieved by improving the completeness of the knowledge base, against which these inferences are validated.

For the *invalid* category, 19 of the 21 (90.47%) distinct inferences were rated as invalid by the raters. Thus as expected, Kilaza performed much better at detecting *invalid* predicates, than it did at identifying *valid* predicates. We can say that Kilaza has a very low error rate of 9.5% (2 out of 21) when detecting the *invalid* inferences on the Professions analogies. This

is surprising, given that Kilaza knows nothing about the target domain of Professions. Thus, it did not use pragmatic or any other factors to detect invalid predicates, using only intra-predicate constraints to achieve this result.

The difference between Kilaza's *valid* and *invalid* categories is also highlighted by the number of predicates that were awarded the lowest rating (of 1) by both raters. A rating of 1 indicates that both raters thought the predicate could not be considered credible under any circumstance (eg `wear pen vote`). These low ratings generally corresponded to "howlers" formed by incongruous combinations of relations and arguments. For the *valid* category, just 25% (10 out of 40) predicates were awarded 1 by both raters, but 67.1% (12 out of 21) *invalid* predicates were awarded the lowest rating by both raters. This indicates that many of Kilaza's *invalid* inferences were agreed by both raters to be completely invalid.

Kilaza's ability to reject clearly invalid predicates is further strengthened when we look at the disagreements between the raters. Just 4.7% (1 out of 21) of the *invalid* predicates were rated differently by the raters (*i.e.* one rater awarded it a valid rating while the other awarded it an invalid rating). However, 30% (12 out of 40) of the *valid* predicates were rated differently by the raters - and were correctly treated as potentially valid predicates by Kilaza. So, Kilaza shows the same variation as people, reflecting the different knowledge that is brought to bear on the validation process.

In conclusion, over 90% of the predicates identified as *invalid* by Kilaza's validation process were agreed by human raters to be invalid. This is perhaps the best reflection of the capability of this validation process on the Professions knowledge base created by Veale (1997).

Adaptation Results and Discussion

This first experiment also examined the ratings given to the adapted predicates. The results of this Experiment on the Adaptation process are summarised in Table 6.3. (Note that all adaptations produced by Kilaza are necessarily in its *valid* category).

		Human Rating		
	Adaptation Accuracy	**Rated Valid**	**Rated Invalid**	**Total**
Kilaza Class	**Valid**	8 (40.0%)	12 (60.0%)	20 (100%)
	Invalid	-	-	-

Table 6. 3 - *Accuracy of the Adaptation process*

All 24 invalid predicates were passed to the adaptation process, and this created 20 new *adapted* predicates - 4 predicates could not be adapted. There were no duplicates among the adapted predicates. Because the original analogies and their inferences were generated by Kilaza, we could not test if the adapted inferences were appropriate adaptations of the inferred information. However, we did test the adapted inferences to see if they were rated as valid predicates.

The average rating awarded to the *adapted* predicates was (M=2.57, SD=1.75). As expected, these average ratings are broadly in line with the predicates from Kilaza's *valid* category above (M=2.62, SD=2.09). When we look at the 20 adapted predicates before and after adaptation, we see that the average ratings have increased from 1.57 (SD=1.23) to 2.57 (SD=1.70). Thus, adaptation has a distinct influence on improving the ratings of the rejected inferences.

Before adaptation, 18 of the 20 (90%) predicates were given invalid ratings by the raters and after adaptation just 12 (60%) were rated as *invalid*. However, this improvement is solely due to the increase in the number of ambiguously rated predicates (*ie* disagreement among the raters), rising from 1 (5%) to 7 (35%) after adaptation. The number of predicates rated as valid both before and after adaptation was 1 (5%). Thus, while adaptation did improve the ratings of rejected inferences, it appears to have had little success at generating unambiguously valid predicates under the given conditions.

These results camouflage one of the problems with the adaptation process. Our validation scheme operates in the absence of pragmatic (and other) factors, so adaptation was effectively performed by finding a list of possible alternatives and selecting the first item from that list. The rejected predicates involved just two relations (`injure` and `wear`), which are both in the `personal-event` class. This resulted in 19 of the 20 adaptations using the `injure` relation, as this is the first adapted relation chosen from `personal-event` class. In a more complete model of adaptation, pragmatic and other information may assist in selecting an appropriate relation from among the alternatives that Kilaza identified.

A detailed inspection of the adapted predicates reveals a potential anomaly that needs to be explained. One predicate that is forwarded to the adaptation process is `(Wear Thermal-Lance Cement)`, but this predicate is also among the predicates that are rejected without adaptation! Examining the original data resolves this apparent contradiction. The rejected predicate was `(Wear713678 Thermal-Lance88015 Cement88589)`, whose `wear` relation originated in the target domain and thus could not be adapted. In contrast, the adapted predicate was `(Wear716794 Thermal-Lance88015 Cement88589)` whose `wear` (note the different numeric-suffix) relation was transferred from the source domain and thus was open to adaptation. So, the apparent anomaly is resolved by identifying which predicates used transferred information, as only this information can be adapted. The same explanation applies to the other two predicates that appear to be both rejected and adapted.

So while adaptation managed to improve the ratings of the 'invalid' predicates, it had little success at generating valid predicates. The most obvious explanation for this is that intra-predicate constraints are too weak to support adaptation in the absence of pragmatic influences.

6.4.3 Conclusions from Experiment 1

The Professions domains represented a significant challenge to the Kilaza validation and adaptation processes because many of the relations and

objects were not contained in the background taxonomy. Thus, the functional attributes required for validation and adaptation were not available.

In summary, the adaptation mechanism had a mild influence on improving the validity of predicates. However, the validation mechanism performed well at detecting *invalid* predicates from the Professions inferences, as the raters agreed that 90.5% of the rejected inferences were indeed invalid. So, validation performed quite accurately - even in the absence of pragmatic and other influences.

6.5 Overview of the Assorted Inferences

The first experiment analysed Kilaza's performance on the Professions collection of semantically similar domains. We conducted a second experiment to examine Kilaza's performance on the smaller Assorted domains described by a much wider variety of relations.

The 81 Assorted domains generated a total of 6561 analogies. From these analogies, Kilaza generated 3793 inferences using its pattern completion model for inference generation. Of these predicates 2158 (56.9%) were classified as *valid* and 1635 (43.1%) inferences were categorised as *invalid* predicates. Thus, the increased use of items from the taxonomy allowed Kilaza to detect more invalid predicates than in the Professions experiment (up from 13.7%). The quantity of inferences in each category is summarised in Table 6.4.

Type of Inference	Number of Inferences
Valid Inferences	2158 (56.9%)
Invalid Inferences	1635 (43.1%)

Table 6. 4 – *Summary of the Assorted Inferences*

Interestingly, 11 of the 2158 (0.5%) validated inferences were validated by identical predicate validation, described as validation mode 1 in Chapter 4. The remaining 2147 (99.5%) of predicates were validated by

functional attribute validation (validation mode 3 in Chapter 4). None of the inferences were validated by partial predicate validation, or validation mode 2 in Chapter 4.

An additional 939 predicates were generated by Kilaza's adaptation process, which was invoked on the *invalid* predicates. So 57.4% (939 out of 1635) of the *invalid* predicates were modified by adaptation. This is a much lower adaptation rate than on the Profession inferences (83.3%), but those adaptations involved a much smaller quantity and variety of predicates.

6.6 Experiment 2 - Validation and the Assorted Domains

We will discuss the accuracy of Kilaza's validation process, before discussing the accuracy of adaptation.

6.6.1 Method

Materials

The materials were taken from the 2159 inferences generated by Kilaza on the Assorted domains. The duplicates were removed from the valid inferences, leaving 1560 distinct *valid* predicates. Two hundred and sixteen of these *valid* inferences were randomly selected to be rated, representing 10% of the original *valid* inferences. Duplicate entries were removed from the *invalid* inferences leaving 542 distinct predicates, from which 50 were randomly selected to be rated, again representing just under 10% of the *invalid* inferences. Of the 651 *adapted* inferences 65 were selected to be rated, and these contained no duplicate predicates. Thus, each rater saw a collection of predicates containing 216 *valid* predicates, 50 *invalid* predicates and 65 *adapted* predicates, presented in random order.

All inferences selected for rating were validated using functional attribute validation (Mode 3).

Participants

As in the previous experiment, two raters were used and both raters were familiar with predicate calculus representation.

Design

As in the previous experiment.

Procedure

As in the previous experiment.

6.6.2 Results and Discussion

Validation (Mode 1) Results and Discussion

The 11 (0.5%) predicates validated by identical predicate validation (Mode 1) received very high ratings. The average rating awarded to these inferences was M=6.55 (SD=0.66), 10 of these inferences were given the highest rating by both raters. The other remaining inference was given ratings of 4 and 6 respectively. Thus as expected, Identical Predicate validation proved a very accurate means of validation, although it only accounted for 0.5% of the total number of generated predicates.

Validation Results and Discussion

The results of this Experiment are summarised in Table 6.5.

		Human Rating		
	Validation Accuracy	**Rated Good**	**Rated Bad**	**Total**
Kilaza Class	**Valid**	113 (52.3%)	103 (47.7%)	216 (100%)
	Invalid	7 (14%)	43 (86%)	50 (100%)

Table 6. 5 - *Accuracy of Validation on the Assorted Inferences*

The average rating awarded to the Assorted inferences that Kilaza categorised as *valid* was M=3.47 (SD=2.66), while the average rating awarded to the *invalid* predicates was M=1.59 (SD=1.52). Thus, predicates in the *invalid* category as expected, received significantly lower ratings that the *valid* predicates. The average rating for the Assorted *valid* inferences is significantly higher than the Professions inferences (M=2.62, SD=2.09). However, the *invalid* inferences are rated similarly on the two collections.

A McNemar's test was also performed to compare Kilaza's classifications to the categorisations awarded by the raters. The results were: #Invalid-RatedGood = 7, #Valid-RatedBad = 103, $p < 0.0001$ showing strong agreement between the two ratings.

Of the 216 predicates selected from Kilaza's *valid* category, 113 (52.3%) were identified as being potentially valid by the human raters, an increase from 42.5% on the Professions results. Thus, 103 (47.7%) of these 216 predicates were rated as invalid. This is a higher rate of detecting invalid predicates than on the Professions inferences (40%). However, it does also indicate that the intra-predicate constraints used by Kilaza to identify invalid predicates can not be expected to detect all cases of invalidity.

For the predicates in Kilaza's *invalid* category, raters agreed that 86% (43 of 50) were correctly categorised as invalid. This is a slight decrease (from 91%) on the accuracy achieved on the Professions inferences. However, the Assorted results are based on a greater number of invalid predicates (50 instead of the Professions 20), using a much wider range of relational predicates (21 instead of the Professions 2).

Kilaza's ability to reject clearly invalid predicates is further strengthened when we look at the disagreements between the raters. 76% (38 out of 50) of the invalid predicates we given the lowest rating (1) by both rates. While raters disagreed on the rating of 25.5% (55 of 216) of the *valid* inferences, they only disagreed about the rating of just 8% (4 of 50) of the *invalid* inferences. This highlights that Kilaza's invalid category identifies clearly invalid predicates.

Adaptation Results and Discussion

The results of this Experiment on *adapting* the Assorted inferences are summarised in Table 6.6.

The average rating awarded to the *adapted* inferences was M=4.20 (SD=2.37). Surprisingly, this is the highest rating given to any category in these experiments. Out of the 65 Assorted adaptations 72.3% (47) were rated as valid by the raters, while only 40% of the Professions inferences received a valid rating. These high ratings are due to an increase in the number of

predicates being rated as valid by both raters. After adapting the Professions inferences, just 5% of the predicates were rated as unambiguously valid, whereas 36.9% of the Assorted adaptations were rated as unambiguously valid. In both the Professions and Assorted domains, raters disagreed about the validity of approximately 35% of the adaptations.

		Human Rating	
	Adaptation Accuracy	**Rated Good**	**Rated Bad**
Kilaza Class	**Valid**	47 (72.3%)	18 (27.7%)
	Invalid	-	-

Table 6. 6 - *Accuracy of the Adaptation process on the Assorted Inferences*

These adaptation results are considerably better than those of the Professions results. This can be attributed to the Assorted KB using domains with relations that were defined by more functional attributes than the Profession inferences. This meant that selecting an alternate predicate could be performed with greater accuracy, because a greater level of detail was available. As with the adaptations from the Professions inferences, we were not able to test if the adapted inferences were appropriate modifications of the original inferences (because the driving analogies were automatically generated).

Unlike the adaptations produced by the Professions domains, adaptations on the Assorted domains made use of a considerably greater diversity of relations. Adapting the Assorted inferences used 15 different relations rather than the 2 relations used to adapt the Professions inferences. So these results are considerably better than the earlier Professions results.

6.6.3 Conclusions from Experiment 2

Overall, the results produced by the Assorted domains were noticeably better than the Professions results, but were still not very successful. Over 52% of the *valid* inferences were correctly accepted by the validation process. Of the

invalid inferences 86% were rated as invalid, even with the greater number and variety of predicates than found in the Professions results. Finally, 72% of the adapted inferences were rated as potentially valid. Again, it must be pointed out that all adaptations were executed without pragmatic or other influences.

Overall, these results showed the improvement expected by using more taxonomically defined items. The use of a greater diversity of more specific relations also had an impact of these results.

6.7 Conclusion from Retrieval and Validation Results

Firstly, if the information used to describe each domain is very detailed, perhaps to the level of individual WordNet synsets for each item, then we can expect the validation process to perform much more accurately. This is because accurately specified relations and objects allow us place more constraints on the internal structure of individual predicates.

Using semantics to identify analogies between such domain descriptions also becomes a more complicated process. With such a huge diversity of relations used to describe various domains, the *direct* predicate identicality constraint would eliminate virtually every analogy. This necessitates identifying the *implicit* identicality between relations, perhaps involving some mechanism of re-representation (Yan *et al*, 2003).

In contrast, using a smaller set of general relations, makes finding novel mappings much easier. Significantly, this also allows us use the standard predicate identicality constraint. However, validating these inferences is less reliable as they place fewer restrictions on their inferences.

6.8 Creativity Tests

In the last chapter we presented a number of retrieval tests that were conducted on creative analogies. These tests focused specifically on Kilaza's ability to discover creative analogical comparisons, including some famous examples like the *solar-system:atom* analogy. As was shown in Section 5.7, the retrieval experiments successfully identified the correct source, which

was among the candidate sources that were retrieved from memory. In this section, we will look at the inferences that were generated from these analogies.

Target	Source	Correct Inference Validated	Number of Inferences Validated
Atom	Solar-System	y	4
Atom-Falkemhainer	Solar-System-Falkenhainer	y	3
General	Surgeon	y	4 (2 unique)
Heat-flow	Water-Flow	y	4 (3 unique)
Leadbelly	Caravaggio	y	4 (1 unique)
Love-triangle	Triangle-Directed	y	0
Requited-love	Love Triangle	y	3 (2 unique)
Fish	Bird	y	4 (3 unique)
Vampire	Banker	y	3 (2 unique)
Cycling	Driving	n	0

Table 6. 7 - *Validating Inferences for the Creative Source Domains*

Table 6.7 summarises the results generated by validating the inferences from these creative analogies. Kilaza generated and validated the correct inferences for 9 (70%) of the creative analogies. (Note that the *love-triangle:directed-triangle* analogy correctly generated no inferences). Almost all of these analogies generated the correct inferences, and these inferences were successfully validated by Kilaza.

One of these analogies also required one inference to be adapted. The `bird:fish` analogy generated the inference `flies-through fish water`, which was correctly adapted to `swim fish water`. Just one of these analogies did not produce the desired inference. Overall however, Kilaza was reasonaly successful in finding these creative analogies.

6.9 Conclusion

This Chapter assessed the usefulness of our post-mapping models, at *validating* inferences and *adapting* invalid inferences. Assessing the validation and adaptation models was based on the inferences generated from the Professions and Assorted collection. This assessment was based on the use of a novel methodology, to compare the results generated by the model to those given by people. The categorisations generated by the Kilaza model were compared to the ratings given by raters in two Experiments - one for each collection.

The Professions inferences (Veale, 1996) tested the usefulness of these models on large domains described by general relations. In contrast the Assorted domains tested the models on smaller domains described by much more specific relations. Invalid inferences were detected with an accuracy of approximately 90% on the two tests. An accurate means of detecting invalid inferences was crucial to our wider objective of making a creative analogising machine. This maximises the creativity of the model as it does not falsely reject too many creative inferences. This is particularly impressive given that validation was performed in the absence of knowledge about the target domain, and about pragmatic and other factors.

Between 42% and 52% of the validated inferences were categorised as valid. This result indicates that intra-predicate validation can play a part in identifying valid inferences, even producing results in the absence of target-specific knowledge. The adaptation results showed that intra-predicate constraints are useful when modifying transferred information, but pragmatic and other information is central to a more complete model of adaptation.

Finally, Kilaza was tested to see if it could discover some of the famous examples of creative analogies, including the *atom:solar-system* analogy and the *heat-flow:water flow* analogy. Kilaza included the correct source domains among the candidate source for each of these analogies. Furthermore, Kilaza also generated the correct inferences for each of these analogies. Therefore Kilaza was successful in its ultimate challenge of finding novel (to Kilaza) analogies for presented target problems.

Chapter 7

Conclusion

"We didn't have metaphors in our day, we didn't beat about the bush"
- - Anon.

7.1 Introduction

We presented the Kilaza model whose goal was to discover novel analogies that supply inferences to a presented target problem. Kilaza was developed as a three-phase model of analogy, encompassing the phases of *retrieval, mapping* and *validation*. The model was designed to maximize its ability to identify novel source domains with which to re-interpret the given target domain, as these domains are strongly associated with creative analogies (Boden, 1992). Its design also attempted to maximize the novelty of the inferences that were supplied to the target, in order to overcome any limitations with the current understanding of the target domain. As discussed in Chapter 2, Kilaza is among the few models that attempt to address three phases of the analogy process, and is the only model to focus on the discovery of novel analogies.

In this chapter we will look back at each model of the three phases. Each review will focus on issues that arise from the development and testing related to that phase model. The chapter ends with some concluding remarks about the project.

7.2 Retrieval and Implications

A number of features distinguish Kilaza from previous models of analogical reasoning. Firstly, Kilaza attempted to overcome the semantic limitations of previous retrieval models, as this tended to reduce the diversity of the source domains that were identified. Instead, Kilaza explored the use of domain's structure as a basis for retrieval, identify domains that are isomorphic and homomorphic to the given target. The structure of the given target is mapped into a separate "structure space", which is an n-dimensional space representing the structure of all source and target domains. Retrieval is performed within this structure space using the nearest-neighbors algorithm.

All phases of the Kilaza model were tested on two different collections of domains. Testing on one collection illustrated that structure has a reliable influence on retrieval, when the domains are described by a small set of relational predicates. That is, domains that are located close to the target domain in structure space tend to form larger inter-domain mappings, while domains located far from the target tend to form smaller mappings. However, this relationship between retrieval and mapping did not translate into the generation of a larger set of inferences. Testing on the second collection of domains (which were described by a much wider range of relational predicates) indicated that increasing the diversity of relations in the domains, effectively eliminated the influence of structure on the retrieval phase. These results highlighted the implicit dependency between the performance of analogy models and the manner in which the problem domains are specified.

One of the most significant factors that was identified during this project is the difference between elaborating a given analogy, and discovering that same analogy afresh. When discovering a novel analogy, an analogy model must be capable of identifying the implicit similarity between the domains. The core difference arises from the fact that known (previously discovered) analogies are described in a manner that makes the inter-domain mapping easy to identify. In contrast, discovering a novel analogy must typically combat the terminology and other differences that can lead to two the two domains being described in quite

different ways. When attempting to understand how novel analogies are discovered afresh, the two domains should be represented in a manner that best describes the understanding of the two domains *before* the analogy was drawn. Only in this way can we come to understand the processes that lead to the "invention" of useful and novel analogies.

7.3 Mapping and Implications

The second of Kilaza's phases performs mapping and inferences generation. While Kilaza is primarily based around a standard incremental mapping model, a number of modifications to the standard algorithm have been made. Firstly Kilaza implements "predicate identicality" as a soft constraint, allowing mappings to be developed between semantically diverse domain descriptions. However, the Kilaza model focused more on the inferences that were mandated by the mapping, rather than on the mapping itself. Developing a novel analogical mapping must (like retrieval) deal with the differences in domain terminology, to identify the implicit commonality between domains. Therefore, Kilaza implements the predicate identicality as a soft constraint.

However, the primary focus of Kilaza was on the inferences rather than on the mapping. Inferences are generated using the standard CWSG algorithm (Holyoak et al, 1994). However, all inferences are sent through a validation process to determine if they should be accepted. Additionally, Kilaza attempts to identify unused source objects, rather than generate the usual Skolem objects in the target domain. This allows Kilaza to generate mappings for a target domain that consists only of isolated objects.

7.4 Validation and Implications

The third novel aspect of Kilaza concerns its validation model. Kilaza attempts to validate inferences in a domain-independent manner, and in the absence of pragmatic or other influences.

The core of the validation process rests on the use of intra-predicate constraints in the form of role restrictions. These served to detect any invalid

inferences before they are introduced to the target domain. (Kilaza also compares inferences to the predicates recorded in memory, however this technique was very ineffective at validating the novel inferences that Kilaza generated).

Kilaza used these invalid inferences to identify source domains that were non-analogous to the target domain. Such analogies were completely rejected by Kilaza. Invalid inferences also served to reject overextensions of an otherwise valid analogy. These analogies and their valid inferences were accepted, while validation prevented the mis-application of some of the source material to the given target.

Kilaza's role restrictions also served to adapt many invalid predicates. However, performing adaptation in the absence of pragmatic factors was insufficient to generate useful adaptations. It is expected that Kilaza may produce much more useful results if pragmatic factors were included in the adaptation process.

7.4.1 Specifity and Generality

The results produced by Kilaza highlight the competing influences of the *specifity* versus the *generality* of the domain descriptions used by an analogy model. We characterize these competing influences by portraying two distinct scenarios under which analogy may be modeled. Firstly under the *generality* scenario, domains are described by a small set of more abstract relational predicates - so as to highlight the generality of the domains information. The use of a small set of general relations enables the structural influence to be used during the retrieval phase. It also allows the predicate identicality constraint to be usefully applied by the mapping model. However, in this scenario, intra-predicate validation becomes less accurate due to the reduced level of detail in the generated inferences. Thus, under the generality scenario, the phases of retrieval and mapping are made simpler, while some of the post-mapping processes may be more difficult – due to the implicit loss of information in the inferred relations.

Secondly in the alternate *specifity* scenario, the situation is quite different. Domains are described by a wide variety of very specific relational predicates, mostly emanating from the lower levels of the taxonomy. As specifity increases (and generality decreases), firstly the predicate identicality constraint looses its effectiveness in both the retrieval and mapping phases. Then, structure becomes an ineffective means for performing analogy retrieval, as structurally similar domains fail to map with the presented target - primarily due to the competing influence of the 1-to-1 mapping constraint. But intra-predicate validation becomes more accurate due to the extra detail available in each inferred predicate, which can be used by the validation model.

7.5 Overall

This work highlighted a number of factors that make discovering a novel analogy a difficult and unpredictable process. Not only is it difficult to find an appropriate source domain, but this source must also be described in such a way as to allow the appropriate mapping and inferences to be generated. For example, Kekulé's analogy between the *carbon-chain* and a *snake* could have been driven by any number of other source domains, from tying a shoe-lace to buckling his belt. Many source domains involve the crucial change from a linear to a ring structure that was central to Kekulé's analogy. However, Kekulé and many others (Gick and Holyoak, 1980) frequently fail to notice these potential analogies.

One of the other crucial factors is how analogy and validation (or adaptation) can rely heavily on target specific intelligence. As highlighted in the first chapter, Kekulé's *sanake:carbon-chain* analogy involved much more creativity than merely accepting the suggested inferences – which might account for the 10 year "delay" in discovering the carbon ring. The ring structure suggested by the analogy had to undergo several successive refinements, before the correct interpretation was derived. In particular, the carbon ring requires that carbon atoms can bond with more than one other carbon atom, which was also an unexpected inference. Each successive refinement required a deep

understanding of the target domain, to assess the implications of the analogy and to resolve any inconsistencies in the resulting interpretation. Of course, this analogy involved mapping the carbon atoms in benzene (the carbon ring) onto the snakes body. Thus 6 *carbon* objects were mapped onto the one *snake* object, which also makes the required mapping more difficult to uncover.

7.6 Future work

Kilaza currently uses its own taxonomy for validation, but greater coverage should be achieved by adopting WordNet to this task. Descriptions of all problem domains could avail of WordNet's detailed information, by annotating all predicates with the relevant WordNet synset. This should help the validation to detect even more invalid inferences, and may even improve the accuracy of this detection process. Of course, using this information would require that all domain descriptions are annotated by the appropriate synset. Introducing WordNet could also allow Kilaza to include a semantic similarity (as well as its identicality preference) component its mapping model.

One item that would greatly assist any future research on finding analogies, is a large collection of domain descriptions. Such a collection should naturally include many analogous domain pairs, as well as many other domains. This would allow testing and comparison of the various phase models, particularly retrieval and validation.

Veale and O'Donoghue (2000) have investigated computational models of conceptual integration networks, or "conceptual blending" (Fauconnier and Turner, 1998). This is seen as encompassing the analogy process within a broader context, in which a solution space is constructed to include the inter-domain mapping as well as items from the two input domains. The intra-predicate constraints used by the Kilaza model may also be applied to the "output" space of conceptual blending, as this space can not be formed from invalid predicates. The intra-predicate constraints used by Kilaza may help to eliminate many of the "incorrect" outputs that might otherwise be created.

7.4 Conclusion

The retrieval model demonstrated that structure can be an influence in retrieving certain types of source domains. The validation model illustrated that achieving validation using a direct comparison to some pre-existing predicates, would require access to a very large number of pre-existing predicates. While such a collection may aid in validating many predicates, it seems unlikely that these will be particularly useful in validating many novel analogical comparisons. The validation results also demonstrated that intra-predicate constraints can be used to reject many of the invalid inferences that are generated when searching for novel analogies.

The creative perspective adopted in this thesis shed some new light on a number of facets of the analogy process. In particular, it highlights the indirect dependency that exists between the retrieval and validation phases of analogy. In summary, Kilaza was successful in its overall goal of finding analogies which are novel to Kilaza and which supply valid inferences to the given target domain.

References

Allen, J. *"Natural Language Understanding",* Benjamin/Cummings, CA, 1995.

Anderson J. R. *"Rules of the Mind",* Lawrence Erlbaum, NJ: 1993.

Anderson J. R. *"The Adaptive Nature of Human Categorisation",* Psychological Review, 98, 409-429, 1991.

Anderson J. R. "*The Architecture of Cognition",* Cambridge, MA: Harvard University Press, 1983.

Barnden J. *"Belief in Metaphor: Taking common sense Psychology Seriously",* Computational Intelligence, 8, 3, 520-552, 1992.

Barsalou, L.W. *"Ad hoc categories",* Memory & Cognition, 11, 211-227, 1983.

Blanchetter I. Dunbar K. *"How Analogies are Generated: The roles of Structural and superficial similarity",* Memory and Cognition, 28, 1, 108-124, 2000.

Bobrow, D. Winograd, T. *"KRL, Another Perspective",* Cognitive Science, vol. 3, 29-42, 1979.

Boden, M. A. *"The Creative Mind",* Abacus, 1992.

Boden, M. A. *"Creativity and Artificial Intelligence",* Artificial Intelligence, 103, 347-356, 1998.

Bohan A. O'Donoghue *"A Model for Geometric Analogies using Attribute Matching"*, AICS-2000 - 11th Artificial Intelligence and Cognitive Science Conference, Aug. 23-25, NUI Galway, Ireland, 2000.

Booch, G. Rumbaugh J. Jackobson I. *"The Unified Modelling Language: Users Guide",* Addison Wesley, 1999.

Bootzin, R. Bower, G. Crocker, J. Hall E. *"Psychology Today: An Introduction",* McGraw-Hill, 1991.

Brachman, R. Schmolze, J. *"An Overview of KL-ONE Knowledge Representation System",* Cognitive Science, 9, 171 - 216, 1985.

Brachman, R. McGuinness, D. Patel-Schneider, P. Resnick, L. and Borgida, **A.** *"Living with CLASSIC: When and How to Use a KL-ONE-Like*

Language", in John Sowa, ed., *Principles of Semantic Networks: Explorations in the representation of knowledge* , Morgan-Kaufmann: San Mateo, California, pages 401--456, 1991.

Brown, A. *"Analogical learning and Transfer: What develops?",* in "Similarity and Analogical Reasoning", Cambridge Univ. Press, 1988.

Brown, M. G. *"A Memory Model of Case Retrieval by Activation Passing",* PhD Thesis, University of Manchester, 1995.

Brown, M.G. *"An Underlying Memory Model to Support Case Retrieval",* Topics in Case Based Reasoning (EW-CBR 93), 132-143, 1994.

Buchanan, B. G., *"Creativity at the Metalevel",* pp 13-28, AAAI Magazine, Vol. 22, 3, Fall 2001.

Bursein M. *"Concept Formation by Incremental Analogical Reasoning and Debugging",* in Michalski, R. Carbonell, J. Mitchell, T. "Machine Learning: An Artificial Intelligence Approach", Volume 2, Morgan Kaufman, CA, 1986.

Catrambone, R. Holyoak K. J. *"Overcoming Contextual Limitations on Problem Solving Transfer",* Journal of Experimental Psychology: Learning, Memory and Cognition, 15, 6, 1147, 1156, 1989.

Clocksin, W. Mellish, C. *"Programming in Prolog",* Springer, 1984.

Copi, I. M. *"Introduction to Logic",* Macmillan, NY, 1982.

Coulson, S. Oakley, T. *"Blending Basics",* Cognitive Linguistics, 11-3/4, 2000.

Crean B, O'Donoghue, D. *"Retrieving Analogs with Features of Structure",* Applied Informatics, Innsbruck, Austria, Feb. 2001.

Crean, B. O'Donoghue D., *"RADAR: Finding Analogies using Attributes of Structure",* LNAI 2464, p. 20-27, AICS 2002, Limerick, 12-13 September 2002.

Crean B, *"Structural Similarity in Analogical Retrieval Models",* Proceedings of 12th AICS, NUI Maynooth 09015 19480, pp 55-64, 2001.

Crean, B. *"Structure-Based Analogy Retrieval",* M.Sc. Thesis, Department of Computer Science, National University of Ireland, Maynooth, Ireland, 2003.

Curie, M. *"Pierre Curie",* translated by C. & V. Kellogg, Macmillan, 1923.

Dunbar K. *"The Analogical Paradox: Why Analogy is so Easy in Naturalistic Settings, Yet so Difficult in the Psychological Laboratory",* in The Analogical Mind, (Ed) Gentner, Holyoak and Kokinov, 2001.

Dunbar, K. Blanchette, I. *"The* in vivo/in vitro *Approach to Cognition: The Case of Analogy",* Trends in Cognitive Sciences, Vol 5, 8, 334-339, 2001.

Duncker, K. *"On Problem Solving",* Psychological Monographs, 58 (whole, no. 270), 1945.

Eliasmith C. and Thagard P. *"Integrating structure and meaning: A distributed model of analogical mapping",* Cognitive Science, 2001

Eskridge, T. C. *"A Hybrid Model of Continuous Analogical Reasoning",* in Analogy, Metaphor and Reminding, Eds. Barnden and Holyoak, Ablex,. Norwood, NJ: 1994.

Evans, T. G. *"A Program for the Solution of a Class of Geometric-Analogy Intelligence Test Questions",* In "Semantic Information Processing", (Ed.) M. Minsky, MIT Press, 1968.

Eysneck M. W. Keane, M. T. *"Cognitive Psychology",* Erlbaum (UK) Taylor & Francis, 3rd Edn. 1995.

Faayad, U. *"Advances in Knowledge Discovery and Data Mining",* AAAI Press, CA, 1996.

Falkenhainer, B. *"An Examination of the Third Stage of the Analogy Process: Verification-Based Analogical Learning",* Proc. IJCAI, 260-263, 1987.

Falkenhainer, B. *"The SME User's Manual, V. 2.0",* Technical Report UIUCDCS-R-88-1421, Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, 1988-a.

Falkenhainer, B. *"Learning from Physical Analogies: A Study in Analogy and the Explanation Process",* PhD Thesis, Univ. Illinois at Urbana-Champaign, 1988-b.

Falkenhainer, B. *"A Unified Approach to Explanation and Theory Formation",* in Shrager, J. & Langley, P. (eds.) Computational Models of Scientific Discovery and Theory Formation, Morgan Kaufman: San Meteo, CA. pp 157-196, 1990.

Falkenhainer, B., Forbus, K. and Gentner, D. *"The Structure-Mapping Engine",* Proceedings of the Fifth National Conference on Artificial Intelligence, 1986.

Falkenhainer, B. Forbus, Gentner, D. *"The Structure Mapping Engine: Algorithm and Examples",* Artificial Intelligence, 41, 1-63, 1989.

Fauconnier, G. Turner, M. *"Conceptual Integration Networks",* Cognitive Science, 22, 2, 1998.

Forbus, K. D. *"Qualitative Process Theory",* Artificial Intelligence, 24, 1, 85-168, 1984.

Forbus, K. "*Exploring analogy in the large",* in Gentner, D., Holyoak, K., and Kokinov, B. (Eds) Analogy: Perspectives from Cognitive Science. MIT Press, 2000.

Forbus K, Ferguson, R. Gentner, D. *"Incremental Structure-Mapping",* Proc. 16th Cognitive Science Society, 313-318, 1994.

Forbus, K. Gentner, D. Law K. *"MAC/FAC: A Model of Similarity-based Retrieval",* Cognitive Science, 19, 2, 141-205, 1995.

Forbus K, Gentner, D. Markman, A. Ferguson, R. *"Analogy Just Looks like High Level Perception: Why a Domain-general Approach to Analogical Mapping is Right",* Journal of experimental and Theoretical Artificial Intelligence, 10, 231-257, 1998.

Forbus K, Oblinger D, *"Making SME Greedy and Pragmatic",* Proc. 12th Cognitive Science Society, 61- 68, 1990.

French R. Hofstadter, D. *"TableTop: A Stochastic, Emergent Model of Analogy-Making",* In Proceedings of Thirteenth Annual Conference of the Cognitive Science Society, pp 708-713, Hillsdale, NJ,: Lawrence Erlbaum, 1991.

Garey, M. and Johnson, D. "*Computers and intractability",* Freeman, San Francisco, 1979.

Gentner, D. *"Structure-Mapping: A Theoretical Framework for Analogy",* Cognitive Science, 7, 155-170, 1983.

Gentner, D. Bowdle B. F. *"The Coherence Imbalance Hypothesis: A Functional Approach to Asymmetry in Comparison",* Proc. Annual Conf. of the Cognitive Science Society, 351-356, 1994.

Gentner, D. Brem S. Ferguson, R. Markman, A. Levidow, B. Wolff, P. Forbus. K. *"Analogical Reasoning and Conceptual Change: A Case Study of Johannes Kepler",* The Journal of the Learning Sciences, 6, 1, 3-40, 1997.

Gentner, D. Forbus, K. *"MAC/FAC: A model of Similarity Based Retrieval",* in Proceedings of Thirteenth Annual Conference of the Cognitive Science Society, Hilldsdale, NJ: Lawrence Erlbaum, 1991.

Gentner, Holyoak and Kokinov (Eds.), *"The Analogical Mind",* MIT Press, 2001.

Gentner, D. Ratterman, M. J. Forbus, K. D. *"The Roles of Similarity in Transfer: Separating Retrievability from Inferential Soundness",* Cognitive Psychology, 25, 524-575, 1993.

Gick, M. Holyoak, K. *"Analogical Problem Solving",* Cognitive Psychology, 12, 306-355, 1980.

Gick, M. Holyoak, K. *"Schema Induction and Analogical Transfer",* Cognitive Psychology, 15, 1-38, 1983.

Gick, M. Paterson, K. *"Do Contrasting Examples Facilitates Schema Acquisition and Analogical Transfer?",* Canadian Journal of Psychology, 46:4, 539-550, 1992.

Goswami, U. Brown, A. *"Melting Chocolate and Melting Snowmen: Analogical Reasoning and Causal Relations",* Cognition, 35, 69-95, 1989.

Goswami, U. *"Analogical Reasoning in Children",* pp 437-470 in "The Analogical Mind", Gentner, Holyoak and Kokinov (Eds.), MIT Press, 2001.

Grice H. Paul *"Logic and Conversation",* in: Syntax and Semantics, Vol. 3, Speech Acts, ed. P. Cole and J. Morgan.New York: Academic Press, 41-58, 1975.

Greiner, R. *"Learning by Understanding Analogies",* Artificial Intelligence, 35, 1, 81-125, 1988.

Grossberg, S. *"A theory of Visual Coding, Memory and Development"* in E. Leewunbrg and J. Buffartt Eds. *Formal Theories of Visual Perception,* Wiley, NY, 1978.

Grossberg, S. *"How Does the Brain Build a Cognitive Code",* Psychological Review, 87, 1, 1-51, 1980.

Hall, R. P. *"Computational Approaches to Analogical Reasoning: A Comparative Analysis",* Artificial Intelligence, 39, 39-120, 1989.

Haykin, S. *"Neural Networks: A Comprehensive Foundation",* Wiley, NY, 1998.

Hoffman, R. *"Monster Analogies",* AI-Magazine, 3, 11-35, 1995.

Hofstadter D. Mitchell, M. *"Conceptual Slippage and Analogy making: A Report on the Copycat Project",* Proceedings 10th Cognitive Science Society, 601 - 607, 1988.

Hofstadter, D. *"Fluid Concepts and Creative Analogies",* Basic Book, NY, 1995.

Holyoak K.J. Hummel, J. *"Analogy in a Physical Symbol System",* Advances in Analogy research, New Bulgaria University, Sofia, 1998.

Holyoak K. J. Koh K. *"Surface and Structural Similarity in Analogical Transfer",* Memory and Cognition, 15, 4, 332 - 340, 1987.

Holyoak K. J. Novick L. Melz E. *"Component Processes in Analogical Transfer: Mapping, Pattern Completion and Adaptation",* in Analogy, Metaphor and Reminding, Eds. Barnden and Holyoak, Ablex, Norwood, NJ: 1994.

Holyoak, K. Gentner, D. Kokinov, B. *"Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational and Neural Sciences",* New Bulgarian University, Sofia, Bulgaria, July, 1998.

Holyoak, K. J. "*The Pragmatics of Analogical Transfer",* Psychology of Learning and Motivation, 19, 59-87, 1985.

Holyoak, K.J. Thagard, P. "*Analogical Mapping by Constraint Satisfaction*", Cognitive Science, 13, 295-355, 1989.

Holyoak, K.J. Thagard, P. *"Mental Leaps: Analogy in Creative Thought",* MIT Press, 1995.

Hummel, J. E. Holyoak, K. J. *"Distributed Representation of Structure: A Theory of Analogical Access and Mapping",* Psychological Review, 104, 3, 427-466, 1997.

Hummel, J. E. Holyoak, K. J. *"LISA: A Computational Model of Analogical Inference and Schema Induction",* Proceedings of 16th Meeting of the Cognitive Science Society, 1996.

Indurkhya, B. *"Metaphor as a Change of Representation",* J. Experimental And Theoretical Artificial Intelligence, 9, 1, 1-36, 1997.

Johnson-Laird P. N. *"Analogy and the Exercise of Creativity",* in Similarity and Analogical Reasoning, Vosniadou, S. Ortony A. (Eds.) Cambridge University Press, 1989.

Jones, R. M., & Langley, P. "*Retrieval and Learning in Analogical Problem Solving"*. In J. D. Moore & J. F. Lehmann (Eds.), Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, pp. 466-471, Hillsdale, NJ: Lawrence Erlbaum, 1995.

Jurafsky, D. Martin J. *"Speech and Language Processing",* Prentice-Hall, 2000.

Just, M. A., Carpenter, P. A. *"A Capacity Theory of Comprehension: Individual Differences in Working Memory",* Psychological Review, 99, 1, 122-149, 1992.

Keane, M. T. *"On drawing analogies when solving problem: A theory and test of solution generation in an analogical problem solving task"*, British Journal of Psychology, 76, 449-458, 1985.

Keane, M. T. *"On Retrieving Analogues when Solving Problems",* Quarterly Journal of Experimental Psychology, 39A , 29-41, 1987.

Keane, M. T. *"Analogical Problem Solving",* Chichester: Ellis Horwood, UK, 1988.

Keane, M. T. *"Incremental Analogising: Theory and Model"* in Gilhooly, K.T. Keane, M.T. Logie R. & Erdos G. (Eds.) Lines of Thinking: reflections on the psychology of thought (Vol. 1) New York: Wiley, 1990.

Keane, M. T. *"On Adaptation in Analogy: Tests of Pragmatic Importance and Adaptability in Analogical Problem Solving"*, Quarterly Journal of Experimental Psychology, 49, (4), 1062-1085, 1996.

Keane, M. T. *"What makes an Analogy Difficult? The Effects of Order and Causal Structure on Analogical Mapping",* Journal of Experimental Psychology: Learning, Memory and Cognition, 23, 946-967, 1997.

Keane, M. T. Brayshaw, M. *"Indirect Analogical Mapping: A Computational Model of Analogy",* in Third European Working Session on Machine Learning. Ed. D. Sleeman, London Pitman, 1988.

Keane, M. T. Ledgeway, T, Duff, S. *"Constraints on Analogical Mapping: A Comparison of Three Models",* Cognitive Science, 18, 387-438, 1994.

Koestler, A. *"Problem Solving"*, 1964.

Kokinov, B. Petrov, A. "*Dynamic extension of episode representation in analogy-making in AMBR"*. In Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society, pp. 274-279, 2000-a.

Kokinov, B. Petrov, A. "*Integrating Memory and Reasoning in Analogy-making: The AMBR model"*. In Gentner, Holyoak, & Kokinov (Eds.), The

Analogical Mind: Perspectives from Cognitive Science, pp. 59-124, Cambridge, MA: MIT Press, 2000-b.

Kokinov, B. *"Analogy is like Cognition: Dynamic, Emergent and Context Sensitive"* In Holyoak Gentner, & Kokinov (Eds.), Advances in Analogy Research, (pp. 96-105), New Bulgarian University, Sofia, Bulgaria, July, 1998.

Kokinov, B. N. *"A Hybrid Model of Reasoning by Analogy",* in Analogy, Metaphor and Reminding, Eds. Barnden and Holyoak, Ablex,. Norwood, NJ: 1994.

Kolodner, J. *"Case Based Reasoning",* Morgan-Kaufman, NY: 1993.

Lakoff, G. *"Women, Fire and Dangerous Things: What Categories Reveal About the Mind",* Chicago Univ. Press, 1987.

Lakoff, G. Johnson, M. *"Metaphors We Live By",* Chicago, Illinois: University of Chicago Press, 1980.

Law, K. Forbus, K. Gentner, D. *"Simulating Similarity-Based Retrieval: A Comparison of ARCS and MAC/FAC",* Proc. 14th Cognitive Science Society, 543-548, 1994.

Lenat, D. *"Eurisko: A Program That Learns New Heuristics and Domain Concepts",* Artificial Ingelligence, 21 (1-2), 61-98, 1983.

Lenz, M. Burkhard H. Brückner S. *"Applying Case Retrieval Nets to Diagnostic Tasks in Technical Domains",* in Advances in Case-Based Reasoning, Eds. Smith, Faltings, Springer, 1996.

Markman, A. *"Constraints on Analogical Inference",* Cognitive Science, 21, 4, 373-418, 1997.

Markman, A. Gentner, D. *"The effects of Alignability on Memory",* Psychological Science, 8, 5, 363-367, 1997.

Markman, A.B., & Gentner, D. *"Structure Mapping in the Comparison Process",* American Journal of Psychology, 113, 501-538, 2000.

Markman, A.B., & Gentner, D. *"Thinking",* Annual Review of Psychology, 52, 223-247, 2001.

Marr, D. *"Vision",* San Francisco: Freeman, 1982.

Mattson, H. F. *"Discrete Mathematics: with Applications",* Wiley Press, 1993.

Michalski, R. Carbonell, J. Mitchell, T. *"Machine Learning: An Artificial Intelligence Approach",* Tioga Press, Palo Alto, CA, 1983.

Miller, A. I. *"Metaphors in Scientific Thought",* Creativity Research Journal, Vol. 9, No 2&3, 113-130, 1996.

Miller G. *"Semantic Networks of English",* Cognition, 41,197-229, 1991.

Miller G. "WordNet: a lexical database for English" in Communications of the ACM, 38, 11, pp. 39 – 41, 1995.

Novick, L. *"Analogical Transfer, Problem Similarity and Expertise",* Journal of Experimental Psychology: Learning, Memory and cognition, 14, 3, 510-520, 1988.

Novick, L. Hemlo *"Transferring Symbolic Representations Across Non-isomorphic Problems",* Journal of Experimental Psychology: Learning, Memory & Cognition, 20, 6, 1296-1321, 1994.

O'Donoghue, D. *"Towards a Computational Model of Creative Reasoning",* Conference on Computational Models of Creative Cognition, Dublin, Ireland, June 30 - July 2 1997.

O'Donoghue, D. *"Constraining Analogical Inference with Memory-based Verification",* Artificial Intelligence and Cognitive Science, UCC, pp 58-64, 1999.

O'Donoghue, D., Winstanley. A. *"Finding Analogous Structures in Cartographic Data",* 4th AGILE Conference on G.I.S. in Europe, Czech Republic, April, 2001.

O'Donoghue, D. Adam Winstanley, Leo Mulhare, Laura Keyes, *"Applications of Cartographic Structure Matching"*, IEEE - IGARSS Conf., July 21-25, Toulouse France, 2003.

O'Donoghue, D. Wyatt, P. *"A Comparison of Convergence Time between sparse Hopfield and Grossberg Neural Networks",* Irish Neural Networks Conference, Maynooth, Ireland, Sept. 1995.

O'Donoghue D. Crean B. *"Searching for Serendipitous Analogies",* European Conference on Artificial Intelligence ECAI - Workshop on Creative Systems, Lyon France, 21-26 July 2002.

Ortony, A. *"Salience, Similes and Asymmetry of Similarity",* Journal of Memory and Language, 24, 569-594, 1985.

Patterson, D. *"Artificial Neural Networks: Theory and Applications",* Prentice-Hall, NY, 1996.

Pazzani, M. *"A Computational Theory of Learning Causal Relationships",* Cognitive Science, 15, 401-424, 1991.

Perkins D. N. Salmon, G. *"Transfer of Learning",* in International Encyclopaedia of Education, Pergamon Press, Oxford, UK, 1992.

Pinker, S. *"The Language Instinct",* Penguin, 1994.

Plate T. A. "Holographic Reduced Representations: Convolution algebra for compositional distributed representations", In John Mylopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence* (IJCAI), pp30-35, Morgan Kaufmann, San Mateo, CA, 1991.

Plate T. A. *"Structured Operations with Distributed Vector Representations",* in "Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational and Neural Sciences", New Bulgarian University, Sofia, Bulgaria, July, 1998.

Plate, T. A. *"Distributed Representations and Nested Compositional Structure",* PhD Thesis, University of Toronto 1994.

Polya, G. *"How to Solve it",* Princeton University Press, 1957.

Reed, S. Ernst, G. Banerji, R. *"The Role of Analogy in Transfer Between Similar Problem States",* Cognitive Psychology, 6, 436-450, 1974.

Reeves, L. Weisberg, R. *"The Role of Content and Abstract Information in Analogical Transfer",* Psychological Bulletin, 115, 3, 381-400, 1994.

Reichgelt, H. *"Knowledge Representation",* Ablex Publishing Corp. NJ, 1991.

Reisbeck, C. Schank, R. *"Inside Case based Reasoning",* Erlbaum, NJ, 1989.

Ritchie G. *"Assessing Creativity",* Proceedings of AISB Symposium on AI and Creativity in Arts and Science, York, March. 2001.

Ross B. H. Kennedy, *"Generalising from the Use of Earlier Examples in Problem Solving",* Journal of Experimental Psychology, 16, 42-55, 1990.

Russell, S. Norvig, P. *"Artificial Intelligence: A Modern Approach",* Prentice Hall, NJ, 1995.

Salvucci D. D. Anderson J. R. *"Integrating Analogical Mapping and General problem solving: the Path-mapping Approach",* Cognitive Science, 25, 67-110, 2001.

Sanders E., Richard J., *"Analogical Transfer as Guided by an Abstraction Process: The Case of Learning by Doing in Text Editing",* Journal of Experimental Psychology: Learning, Memory and Cognition, Vol. 23, No6, 1459-1483, 1997.

Schank, Roger C. *"Dynamic Memory: A theory of reminding and learning in computers and people",* Cambridge, England: Cambridge University Press, 1982.

Shastri, L. Ajjanagadde, V. *"From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables, and Dynamic Bindings using Temporal Synchrony*", Behavioral and Brain Sciences, 16, 259-310, 1993.

Shelley, C., *"Multiple Analogies in Evolutionary Biology",* Studies in History, Philosophy, Biology and Biomedical Science, 30, 2, pp 143-180, 1999.

Sowa J. F. "*Principles of Semantic Networks",* Morgan Kaufman, US, 1992.

Sowa, John F., & Arun K. Majumdar *"Analogical Reasoning"* in de Moor *et al.* International Conference on Conceptual Structures in Dresden, Germany, pp. 16-36, July 2003.

Spellman, B. Holyoak K. J. *"Pragmatics in Analogical Mapping",* Cognitive Psychology, 31, 307-346, 1996.

Spiro, R. Feltovich, P. Coulson R. Anderson, D. *"Multiple Analogies for Complex Concepts: Antidotes for Analogy-Induced Misconception in Advanced Knowledge Acquisition",* pp 498-531 in Similarity and Analogical Reasoning, Vosniadou, S. Ortony A. (Eds.) Cambridge University Press, 1989.

Thagard, P. Holyoak K. J. Nelson, G. Gochfeld, D. *"Analogue Retrieval by Constraint Satisfaction",* Artificial Intelligence, 46, 259-10, 1990.

Thomason, R, Touretzky, D. *"Inheritance Theory and Networks with Roles",* in Principles of Semantic Networks, (Ed.) Sowa J., Morgan Kaufman, CA, 231-266, 1992.

Touretzky, D. ***"****The Mathematics of Inheritance Systems",* Morgan Kaufman, CA, 1986.

Touretzky, D. Thomason, R. Horty, J. *"A Skeptic's Menagerie: Conflictors. Preemptors, Reinstaters and Zombies and non-monotonic inheritance",* In J. Mylopoulos and R. Reiter, eds., Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, Los Altos, pp. 478-483, 1991.

Turner, M. Fauconnier, G. *"A Mechanism of Creativity",* Poetics Today, Volume 20, No. 3, pages 397-418, 1999.

Tversky, A. *"Features of Similarity",* Psychological Review, 84, 327-352, 1977.

Veale, T. *"Metaphor, Memory and Meaning",* PhD Thesis, Trinity College, Dublin, 1995.

Veale, T. Keane, M. T. *"The Competence of Sub-Optimal Theories of Mapping on Hard Problems",* Proc. Fifteenth I. J. C. A. I., Nagoya, Japan, 1997.

Veale, T. O'Donoghue, D. Keane, M. T. *"Computability as the Ultimate Cognitive Constraint",* Annual Conference of the International Cognitive Linguistics Association, Albuquerque: NM, 1995.

Veale T., O'Donoghue D., *"Computation and Blending",* Cognitive Linguistics (special Issue on Conceptual Blending), Vol. 11, Issue 3/4, Pages 253-281, 2000.

Veale, T. Smyth, B. O'Donoghue, D. Keane, M. T. *"Representational Myopia in Cognitive Mapping",* AAAI-96 Computational Cognitive Modelling Workshop, Portland: OE, 1996.

Vosniadou, S. Ortony, A. *"Similarity and Analogical Reasoning",* Cambridge Univ. Press, 1988.

Wasserman, P. D. Oetzel R. M. *"NeuralSource: A Bibliographic Guide to Artificial Neural Networks",* Van Nostrand, NY, 1990.

Weiner, E. J. *"A Knowledge Representation Approach to Understanding Metaphors",* Computational Linguistics, 10, 1, 1984.

Wilson J. D. Buffa, A. J. *"College Physics",* Prentice-Hall (3rd Edition), 1997.

Winston, P. H. *"Learning and Reasoning by Analogy",* Communications of the ACM, 23, 12, 689-703, 1980.

Wolff, P. Gentner, D. *"Evidence for Role-Neutral Initial Processing of Metaphors",* Journal of Experimental Psychology: Learning, Memory and Cognition, 26, No. 2, 529-541, 2000.

Wolverton M. *"An Investigation of Marker-Passing Algorithms for Analogue Retrieval",* Lecture Notes in Artificial Intelligence, 359-370, 1995.

Woods W. *"Understanding Subsumption and Taxonomy: A Framework for Progress.",* In Principles of Semantic Networks, Morgan Kaufman, US, 1991.

Woods W. *"What's in a Link: Foundations for Semantic Networks. Representation and Understanding",* Representation and understanding: Studies in Cognitive Science Academic Press, NY, 1975.

Yan, J. Forbus, K.D. Gentner, D. *"A Theory of Rerepresentation in Analogical Matching",* Proceedings of the 25th Annual Cognitive Science Society, Boston MA, July 31-Aug 2, 2003.

Appendix A

Professions Domains and Assorted domains collections.

;;;ACME Professions Domains

```
BUTCHER
        (DEPEND         PERSON          PERSONAL-HEALTH)
        (PART           FAMILY          FAMILY-RELATIVE)
        (PART           GENE-POOL       CHARACTERISTIC)
        (PART           GENE-POOL       GENE)
        (DEPEND         FAMILY          GENE-POOL)
        (PART           FAMILY-TREE     FAMILY-RELATIVE)
        (DEPEND         FAMILY-TREE     FAMILY-BREEDING)
        (DEPEND         FAMILY          FAMILY-TREE)
        (DEPEND         PERSON          FAMILY)
        (LOCATION-OF    BUTCHER         ABATOIRE); "location" changed to Location-of
        (AFFECT         BUTCHER         LIVESTOCK)
        (AFFECT         BUTCHER         CARCASS)
        (AFFECT         BUTCHER         MEAT)
        (PART           SLAUGHTER       CLEAVER)
        (AFFECT         SLAUGHTER       CARCASS)
        (EFFECT         SLAUGHTER       BLOOD)
        (EFFECT         SLAUGHTER       DEAD)
        (EFFECT         SLAUGHTER       PAIN)
        (EFFECT         SLAUGHTER       FEAR)
        (PERFORM        BUTCHER         SLAUGHTER)
        (PART           CARCASS         TORSO)
        (PART           CARCASS         ARM)
        (PART           CARCASS         LEG)
        (PART           CARCASS         HEAD)
        (PART           LIVESTOCK       CARCASS)
        (PART           LIVESTOCK       COW)
        (CONTROL        BUTCHER         LIVESTOCK)
        (CONTROL        BUTCHER         CLEAVER)
        (WEAR           BUTCHER         WHITE-APRON))))

GENERAL
        (PART           BATTLEFIELD     TRENCH)
        (PART           BATTLEFIELD     CASUALTY)
        (PART           CORPSE          TORSO)
        (PART           CORPSE          ARM)
        (PART           CORPSE          LEG)
        (PART           CORPSE          HEAD)
        (PART           BATTLEFIELD     CORPSE)
        (LOCATION-OF    GENERAL         BATTLEFIELD)
        (CONTROL        GENERAL         ARMY)
        (AFFECT         NERVE-GAS       ENEMY-ARMY)
        (AFFECT         NERVE-GAS       ENEMY-SOLDIER)
        (CONTROL        GENERAL         NERVE-GAS)
        (PART           ATOMIC-BOMB     URANIUM)
        (PART           ATOMIC-BOMB     RADIATION)
        (EFFECT         ATOMIC-BOMB     RADIOACTIVITY)
        (AFFECT         ATOMIC-BOMB     ENEMY-ARMY)
        (CONTROL        GENERAL         ATOMIC-BOMB)
        (CONTROL        COMMAND-CENTRE  INFORMATION-FLOW)
        (CONTROL        COMMAND-CENTRE  INTELLIGENCE)
        (CONTROL        GENERAL         COMMAND-CENTRE)
        (CONTROL        GENERAL         SNUB-FIGHTER)
        (CONTROL        GENERAL         BOMBER-PLANE)
        (CONTROL        GENERAL         SOLDIER)
        (DEPEND         GENERAL         ARMY)
        (CREATE         GENERAL         MILITARY-PROPAGANDA)
        (CREATE         GENERAL         PLAN)
        (AFFECT         GENERAL         ARMY)
        (PART           ENEMY-ARMY      ENEMY-SOLDIER)
        (AFFECT         GENERAL         ENEMY-ARMY)
        (AFFECT         GENERAL         ENEMY-SOLDIER)
        (AFFECT         GENERAL         SOLDIER)
        (AFFECT         BOMBING-RAID    SOLDIER)
        (AFFECT         BOMBING-RAID    CASUALTY)
```

(EFFECT BOMBING-RAID BLOOD)
(EFFECT BOMBING-RAID DEAD)
(PART BOMBING-RAID HEAT-SEEKER)
(PART BOMBING-RAID SMART-BOMB)
(PART BOMBING-RAID BOMB)
(PART BOMBING-RAID MISSILE)
(CONTROL SNUB-FIGHTER MISSILE)
(CONTROL SNUB-FIGHTER BOMB)
(PART BOMBING-RAID SNUB-FIGHTER)
(CONTROL BOMBER-PLANE BOMB)
(CONTROL BOMBER-PLANE MISSILE)
(PART BOMBING-RAID BOMBER-PLANE)
(PERFORM GENERAL BOMBING-RAID)
(WEAR GENERAL MILITARY-UNIFORM)
(CONTROL 18TH-CENTURY-GENERAL MUSKET)
(CONTROL 18TH-CENTURY-GENERAL SWORD)
(CONTROL 18TH-CENTURY-GENERAL ARMY)
(CONTROL 18TH-CENTURY-GENERAL SOLDIER)
(DEPEND 18TH-CENTURY-GENERAL ARMY)
(IS-A-MINUS MILITARY-PROPAGANDA FACTUAL)
(CREATE 18TH-CENTURY-GENERAL MILITARY-PROPAGANDA)
(PART PLAN EVENT)
(PART PLAN SCENARIO)
(CREATE 18TH-CENTURY-GENERAL PLAN)
(AFFECT 18TH-CENTURY-GENERAL SOLDIER)
(AFFECT MILITARY-MANOEVRE CASUALTY)
(PART MILITARY-MANOEVRE WEAPON)
(PART MILITARY-MANOEVRE SOLDIER)
(EFFECT MILITARY-MANOEVRE BLOOD)
(EFFECT MILITARY-MANOEVRE DEAD)
(AFFECT CAVALRY-CHARGE SOLDIER)
(PART CASUALTY TORSO)
(PART CASUALTY ARM)
(PART CASUALTY LEG)
(PART CASUALTY HEAD)
(AFFECT CAVALRY-CHARGE CASUALTY)
(EFFECT CAVALRY-CHARGE BLOOD)
(EFFECT CAVALRY-CHARGE DEAD)
(PART CAVALRY-CHARGE SWORD)
(PART CAVALRY-CHARGE MUSKET)
(PART CAVALRY-CHARGE CANNON)
(PERFORM 18TH-CENTURY-GENERAL CAVALRY-CHARGE)
(WEAR 18TH-CENTURY-GENERAL MILITARY-UNIFORM)
(CONTROL GEORGE-PATTEN SWORD)
(PART ARTILLERY CANNON)
(PART ARTILLERY SOLDIER)
(PART ARMY ARTILLERY)
(PART SOLDIER TORSO)
(PART SOLDIER ARM)
(PART SOLDIER LEG)
(PART SOLDIER HEAD)
(SUBSTANCE MEDAL METAL)
(PART MILITARY-UNIFORM MEDAL)
(WEAR SOLDIER MILITARY-UNIFORM)
(PART ARMY SOLDIER)
(CONTROL GEORGE-PATTEN ARMY)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(PART CANNON CANNON-BALL)
(PART NAPOLEONIC-RUSSIAN-CAMPAIGN CANNON)
(PERFORM NAPOLEON NAPOLEONIC-RUSSIAN-CAMPAIGN)
(SUBSTANCE SWORD STEEL)
(CONTROL NAPOLEON SWORD))))

POLITICIAN
(AFFECT POLITICIAN ELECTORATE)
(AFFECT POLITICIAN HISTORY)
(AFFECT POLITICIAN SOCIETY)

(CREATE POLITICIAN SOCIETY)
(PART POLITICAL-RHETORIC FACT)
(PART POLITICAL-RHETORIC LIE)
(PART POLITICAL-RHETORIC PROMISE)
(CREATE POLITICIAN POLITICAL-RHETORIC)
(CREATE POLITICIAN POLITICAL-MANIFESTO)
(AFFECT LEGAL-LAW SOCIETY)
(CREATE POLITICIAN LEGAL-LAW)
(DEPEND POLITICIAN POLITICAL-MANIFESTO)
(DEPEND POLITICIAN ELECTORATE)
(PART POLICE-FORCE POLICEMAN)
(CONTROL POLITICIAN POLICE-FORCE)
(PART GOVERNMENT CIVIL-SERVANT)
(CONTROL POLITICIAN GOVERNMENT)
(CONTROL POLITICIAN SOCIETY)
(PART ELECTORATE VOTER)
(CONTROL POLITICIAN ELECTORATE)
(PERFORM POLITICIAN OATH-OF-OFFICE)
(PERFORM POLITICIAN POLITICAL-IDEOLOGY)
(PERFORM POLITICIAN SOCIAL-ENGINEERING)
(CREATE VLADIMIR-LENIN COMMUNISM)
(AFFECT VLADIMIR-LENIN 20TH-CENTURY)
(AFFECT VLADIMIR-LENIN SOCIETY)
(AFFECT POLITICAL-LEADER HISTORY)
(PART VOTER VOTE)
(PART SOCIETY VOTER)
(PART SOCIETY CITIZEN)
(AFFECT POLITICAL-LEADER SOCIETY)
(PERFORM POLITICAL-LEADER POLITICAL-IDEOLOGY)
(PERFORM POLITICAL-LEADER SOCIAL-ENGINEERING)
(EFFECT NAZISM INJUSTICE)
(EFFECT NAZISM MURDER)
(CREATE ADOLF-HITLER NAZISM)
(PERFORM LIBERAL-POLITICIAN LIBERALISM)
(PERFORM CONSERVATIVE-POLITICIAN CONSERVATISM)
(SUBSTANCE MAN HUMAN-BONE)
(SUBSTANCE MAN HUMAN-FLESH)
(PART HUMAN-HAIR HAIR-FOLLICLE)
(PART MAN HUMAN-HAIR)
(PART HUMAN-SKIN HAIR-FOLLICLE)
(PART HUMAN-SKIN PORE)
(PART MAN HUMAN-SKIN)
(PART MAN HAND)
(PART EAR EAR-LOBE)
(LOCATION-OF EAR HEAD)
(PART MAN EAR)
(PART MAN NOSE)
(PART MAN EYE)
(PART MOUTH TEETH)
(LOCATION-OF MOUTH FACE)
(PART FACE MOUTH)
(LOCATION-OF NOSTRIL NOSE)
(LOCATION-OF NOSTRIL FACE)
(PART NOSE NOSTRIL)
(LOCATION-OF NOSE FACE)
(PART FACE NOSE)
(LOCATION-OF EYE FACE)
(PART FACE EYE)
(LOCATION-OF FACE HEAD)
(PART MAN FACE)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY))))

SCIENTIST
(CREATE SCIENTIST SCIENCE-PRESENTATION)
(CREATE SCIENTIST SCIENTIFIC-LAW)
(PART SCIENTIFIC-THEORY SCIENTIFIC-LAW)
(PART SCIENTIFIC-THEORY AXIOM)

(CREATE SCIENTIST SCIENTIFIC-THEORY)
(PART MATHEMATICAL-MODEL RULE)
(PART MATHEMATICAL-MODEL AXIOM)
(CREATE SCIENTIST MATHEMATICAL-MODEL)
(CREATE SCIENTIST SCIENTIFIC-LITERATURE)
(DEPEND SCIENTIST SCIENTIFIC-LAW)
(DEPEND SCIENTIST GOD)
(DEPEND SCIENTIST SCIENTIFIC-LITERATURE)
(PART SCIENCE-PRESENTATION FACT)
(PERFORM SCIENTIST SCIENCE-PRESENTATION)
(PART SCIENTIFIC-METHOD EXPERIMENT)
(AFFECT SCIENTIFIC-LAW SOCIETY)
(PART SCIENTIFIC-METHOD SCIENTIFIC-LAW)
(PART SCIENTIFIC-METHOD SCIENTIFIC-PRINCIPLE)
(PERFORM SCIENTIST SCIENTIFIC-METHOD)
(PART EXPERIMENT LASER-MIRROR)
(PART EXPERIMENT LABORATORY-RAT)
(LOCATION-OF EXPERIMENT LABORATORY)
(PERFORM SCIENTIST EXPERIMENT)
(PART LABORATORY-RAT LEG)
(PART LABORATORY-RAT HEAD)
(AFFECT SCIENTIST LABORATORY-RAT)
(PART VOTER VOTE)
(PART SOCIETY VOTER)
(PART SOCIETY CITIZEN)
(AFFECT SCIENTIST SOCIETY)
(SUBSTANCE SURGICAL-GLOVE RUBBER)
(PART SCIENTIST SURGICAL-GLOVE)
(WEAR SCIENTIST WHITE-SMOCK)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(CONTROL SCIENTIST LABORATORY-ASSISTANT))))

ARCHITECT
(PART TRAFFIC-SYSTEM ROAD)
(AFFECT ARCHITECT TRAFFIC-SYSTEM)
(AFFECT ARCHITECT OCCUPANT)
(AFFECT ARCHITECT HISTORY)
(PART VOTER VOTE)
(PART SOCIETY VOTER)
(PART SOCIETY CITIZEN)
(AFFECT ARCHITECT SOCIETY)
(CREATE ARCHITECT BLUEPRINT)
(CREATE ARCHITECT EDIFACE)
(PART CITY TRAFFIC-NETWORK)
(PART CITY OCCUPANT)
(PART CITY CITIZEN)
(PART CITY EDIFACE)
(CREATE ARCHITECT CITY)
(PART ARCHITECTURAL-MODEL PLASTIC)
(CREATE ARCHITECT ARCHITECTURAL-MODEL)
(DEPEND ARCHITECT ZONING-REGULATION)
(PART BUILDER OVERALLS)
(PART CONSTRUCTION-CREW BUILDER)
(PART CONSTRUCTION LADDER)
(PART CONSTRUCTION PILE-DRIVER)
(PART CONSTRUCTION CEMENT-MIXER)
(PERFORM CONSTRUCTION-CREW CONSTRUCTION)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(PART EDIFACE OCCUPANT)

(PART EDIFACE BLUEPRINT)
(PART EDIFACE FLOOR)
(PART EDIFACE CEMENT)
(PART EDIFACE MORTAR)
(PART EDIFACE BEAM)
(PART EDIFACE BRICK)
(CREATE CONSTRUCTION-CREW EDIFACE)
(CONTROL ARCHITECT CONSTRUCTION-CREW)
(PERFORM ARCHITECT SCHOOL-OF-ARCHITECTURE)
(PERFORM ARCHITECT CIVIL-ENGINEERING)
(CREATE FRANK-LLOYD-WRIGHT GUGGENHEIM-MUSEUM)
(CREATE FRANK-LLOYD-WRIGHT MODERNISM)
(CREATE WALTER-GROPIUS BAUHAUS)
(PERFORM MODERNIST-ARCHITECT MODERNISM)
(PART SCHOOL-OF-ARCHITECTURE FIGUREHEAD)
(PERFORM GOTHIC-ARCHITECT GOTHIC-ARCHITECTURE))))

PRIEST
(DEPEND PRIEST REVELATION)
(DEPEND PRIEST CONGREGATION)
(DEPEND PRIEST BIBLE)
(DEPEND PRIEST GOD)
(PERFORM PRIEST SERMON)
(EFFECT TRANSUBSTANTIATION CONVERSION)
(AFFECT TRANSUBSTANTIATION WATER)
(CREATE TRANSUBSTANTIATION WINE)
(PERFORM PRIEST TRANSUBSTANTIATION)
(PERFORM PRIEST VOW-OF-CHASTITY)
(PERFORM PRIEST RITUAL)
(PART RELIGION MIRACLE)
(PART RELIGION MORAL-PRINCIPLE)
(PART RELIGION RITUAL)
(PART RELIGION COMMANDMENT)
(DEPEND RELIGION GOD-HEAD)
(DEPEND RELIGION PRAYER)
(DEPEND RELIGION FAITH)
(DEPEND RELIGION CONGREGATION)
(PART RELIGIOUS-TEXT PARABLE)
(SUBSTANCE CABALIC-MESSAGE HERMENUETIC-CODE)
(PART MORAL-DIRECTIVE CABALIC-MESSAGE)
(PART RELIGIOUS-TEXT MORAL-DIRECTIVE)
(PART RELIGION RELIGIOUS-TEXT)
(PERFORM PRIEST RELIGION)
(CREATE PRIEST SERMON)
(PART RITUAL SACRIFICIAL-LAMB)
(PART RITUAL ALTER)
(LOCATION-OF RITUAL CHURCH)
(PART CHURCH ALTER)
(LOCATION-OF PRIEST CHURCH)
(AFFECT PRIEST HERETIC)
(AFFECT PRIEST CONGREGATION)
(PART CANNON CANNON-BALL)
(PART ARTILLERY CANNON)
(PART ARTILLERY SOLDIER)
(PART ARMY ARTILLERY)
(PART SOLDIER TORSO)
(PART SOLDIER ARM)
(PART SOLDIER LEG)
(PART SOLDIER HEAD)
(SUBSTANCE MEDAL METAL)
(PART MILITARY-UNIFORM MEDAL)
(WEAR SOLDIER MILITARY-UNIFORM)
(PART ARMY SOLDIER)
(PART ARMY-OF-DARKNESS HERETIC)
(AFFECT PRIEST ARMY-OF-DARKNESS)
(AFFECT PRIEST BELIEVER)
(AFFECT PRIEST SOUL)
(AFFECT PRIEST SACRIFICIAL-LAMB)
(PART BLACK-ROBE CRUCIFIX)
(WEAR PRIEST BLACK-ROBE)
(CONTROL PRIEST HOLY-WATER)
(SUBSTANCE CRUCIFIX METAL)
(CONTROL PRIEST CRUCIFIX)
(PART CONGREGATION BELIEVER)
(PERFORM CONGREGATION HYMN-SINGING)

(DEPEND SENSE-OF-BELONGING FAITH)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(PART BELIEVER DOUBT)
(PART BELIEVER FAITH)
(DEPEND SENSE-OF-BELONGING BELIEVER)
(DEPEND RELIGIOUS-FAITH SENSE-OF-BELONGING)
(DEPEND CONGREGATION RELIGIOUS-FAITH)
(CONTROL PRIEST CONGREGATION))))

CHEF
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(AFFECT CHEF DINER)
(AFFECT CHEF MEAT)
(CONTROL CHEF KETCHUP)
(CONTROL CHEF CARVING-KNIFE)
(PART RECIPE INGREDIENT)
(CREATE CHEF RECIPE)
(PART WEDDING-CAKE FLOUR)
(PART WEDDING-CAKE CREAM)
(PART WEDDING-CAKE EGG)
(PART WEDDING-CAKE INGREDIENT)
(PART WEDDING-CAKE CAKE-TIER)
(PART WEDDING-CAKE ICING)
(CREATE CHEF WEDDING-CAKE)
(CREATE CHEF CAKE)
(CREATE CHEF DESSERT)
(PERFORM CHEF COOKING-STYLE)
(AFFECT SAUCE MEAT)
(PART SAUCE-MARINATION SAUCE)
(PERFORM CHEF SAUCE-MARINATION)
(PART BAKERY PREPARATION)
(EFFECT BAKERY DESSERT)
(PART CAKE INGREDIENT)
(PART CAKE FLOUR)
(PART CAKE CREAM)
(PART CAKE EGG)
(EFFECT BAKERY CAKE)
(PERFORM CHEF BAKERY)
(PART COOKERY PREPARATION)
(EFFECT COOKERY DINNER)
(PERFORM CHEF COOKERY)
(SUBSTANCE DINNER-TABLE WOOD)
(PART KITCHEN DINNER-TABLE)
(SUBSTANCE KITCHEN-TABLE WOOD)
(PART KITCHEN KITCHEN-TABLE)
(PART KITCHEN FOOD)
(PART KITCHEN CARVING-KNIFE)
(LOCATION-OF CHEF KITCHEN)
(WEAR CHEF WHITE-APRON)
(PART COOKING-STYLE FIGUREHEAD)
(PERFORM NOUVELLE-CHEF NOUVELLE-CUISINE))))

COMPOSER-professions
(AFFECT COMPOSER LISTENER)
(PART THEATRE CURTAIN)
(PART THEATRE STAGE)
(LOCATION-OF COMPOSER THEATRE)
(DEPEND COMPOSER INSPIRATION)
(DEPEND COMPOSER ORCHESTRA)

(PART LISTENERSHIP LISTENER)
(DEPEND COMPOSER LISTENERSHIP)
(CONTROL COMPOSER CONDUCTOR-BATON)
(PART PERCUSSION DRUM)
(PART PERCUSSION MUSICIAN)
(PART ORCHESTRA PERCUSSION)
(PART ORCHESTRA WOOD-WIND)
(PART ORCHESTRA MUSICIAN)
(PART MUSIC-RECITAL PIANO)
(PART MUSIC-RECITAL VIOLIN)
(PART MUSIC-RECITAL MUSIC-ORGAN)
(PART MUSIC-RECITAL DRUM)
(PERFORM ORCHESTRA MUSIC-RECITAL)
(CONTROL COMPOSER ORCHESTRA)
(CREATE COMPOSER LIBRETTO)
(CREATE COMPOSER MUSICAL-SCORE)
(PART OPERA OPERATIC-ACT)
(PART OPERA CHARACTER)
(PART OPERA MUSIC-NOTE)
(SUBSTANCE MUSICAL-SCORE PAPER)
(PART MUSICAL-SCORE MUSIC-NOTE)
(PART OPERA MUSICAL-SCORE)
(PART LIBRETTO EVENT)
(PART LIBRETTO SCENARIO)
(PART LIBRETTO CHARACTER)
(PART OPERA LIBRETTO)
(EFFECT OPERA JOY)
(CREATE COMPOSER OPERA)
(PART SYMPHONY CHARACTER)
(PART SYMPHONY MUSIC-NOTE)
(PART SYMPHONY SYMPHONIC-MOVEMENT)
(CREATE COMPOSER SYMPHONY)
(PART MUSIC-COMPOSITION PIANO)
(PERFORM COMPOSER MUSIC-COMPOSITION)
(PERFORM TCHAIKOVSKY 1812-OVERTURE)
(PART 1812-OVERTURE DRUM)
(CREATE TCHAIKOVSKY 1812-OVERTURE)
(CONTROL TCHAIKOVSKY CONDUCTOR-BATON)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(PERFORM WAGNER PARSIFAL)
(CREATE WAGNER PARSIFAL)
(CONTROL WAGNER CONDUCTOR-BATON))))

AUTHOR
(CONTROL AUTHOR TYPEWRITER)
(CONTROL AUTHOR READER)
(CONTROL AUTHOR READERSHIP)
(CONTROL AUTHOR PEN)
(DEPEND AUTHOR WRITERS-BLOCK)
(DEPEND AUTHOR IMAGINATION)
(DEPEND AUTHOR READERSHIP)
(CREATE AUTHOR IDEA)
(CREATE AUTHOR PLOT)
(CREATE AUTHOR NOVEL)
(PERFORM AUTHOR CREATIVE-WRITING)
(AFFECT AUTHOR NOVEL)
(AFFECT AUTHOR READER)
(AFFECT AUTHOR SOCIETY)
(AFFECT AUTHOR HISTORY)
(CONTROL ROMANTIC-AUTHOR TYPEWRITER)
(CONTROL ROMANTIC-AUTHOR READER)
(CONTROL ROMANTIC-AUTHOR READERSHIP)
(CONTROL ROMANTIC-AUTHOR PEN)
(DEPEND ROMANTIC-AUTHOR IMAGINATION)
(DEPEND ROMANTIC-AUTHOR READERSHIP)
(CREATE ROMANTIC-AUTHOR IDEA)
(CREATE ROMANTIC-AUTHOR PLOT)

(CREATE ROMANTIC-AUTHOR NOVEL)
(PERFORM ROMANTIC-AUTHOR CREATIVE-WRITING)
(AFFECT ROMANTIC-AUTHOR NOVEL)
(AFFECT ROMANTIC-AUTHOR PLOT)
(AFFECT ROMANTIC-AUTHOR SOCIETY)
(AFFECT ROMANTIC-AUTHOR HISTORY)
(CONTROL GOTHIC-AUTHOR TYPEWRITER)
(CONTROL GOTHIC-AUTHOR READER)
(CONTROL GOTHIC-AUTHOR READERSHIP)
(CONTROL GOTHIC-AUTHOR PEN)
(DEPEND GOTHIC-AUTHOR IMAGINATION)
(DEPEND GOTHIC-AUTHOR READERSHIP)
(CREATE GOTHIC-AUTHOR HORROR)
(CREATE GOTHIC-AUTHOR IDEA)
(CREATE GOTHIC-AUTHOR PLOT)
(CREATE GOTHIC-AUTHOR NOVEL)
(PERFORM GOTHIC-AUTHOR CREATIVE-WRITING)
(AFFECT GOTHIC-AUTHOR NOVEL)
(AFFECT GOTHIC-AUTHOR SOCIETY)
(AFFECT GOTHIC-AUTHOR HISTORY)
(DEPEND MIND MIND)
(DEPEND MIND RATIONALITY)
(DEPEND MIND THOUGHT)
(DEPEND MIND INTELLIGENCE)
(AFFECT SELF-HELP-AUTHOR MIND)
(CONTROL SELF-HELP-AUTHOR TYPEWRITER)
(CONTROL SELF-HELP-AUTHOR READER)
(CONTROL SELF-HELP-AUTHOR READERSHIP)
(CONTROL SELF-HELP-AUTHOR PEN)
(DEPEND SELF-HELP-AUTHOR IMAGINATION)
(DEPEND SELF-HELP-AUTHOR READERSHIP)
(CREATE SELF-HELP-AUTHOR IDEA)
(CREATE SELF-HELP-AUTHOR PLOT)
(CREATE SELF-HELP-AUTHOR NOVEL)
(PERFORM SELF-HELP-AUTHOR CREATIVE-WRITING)
(AFFECT SELF-HELP-AUTHOR NOVEL)
(AFFECT SELF-HELP-AUTHOR PLOT)
(CONTROL NORMAN-MAILER TYPEWRITER)
(CONTROL NORMAN-MAILER READER)
(CONTROL NORMAN-MAILER READERSHIP)
(CONTROL NORMAN-MAILER PEN)
(EFFECT WRITERS-BLOCK DISCOMFORT)
(EFFECT WRITERS-BLOCK INADEQUACY)
(EFFECT WRITERS-BLOCK STOP)
(DEPEND NORMAN-MAILER WRITERS-BLOCK)
(DEPEND NORMAN-MAILER IMAGINATION)
(PART READERSHIP READER)
(DEPEND NORMAN-MAILER READERSHIP)
(CREATE NORMAN-MAILER IDEA)
(CREATE NORMAN-MAILER PLOT)
(CREATE NORMAN-MAILER NOVEL)
(PART TYPEWRITER CARRIAGE-RETURN)
(PART TYPEWRITER KEYBOARD)
(PART CREATIVE-WRITING TYPEWRITER)
(PART PEN INK)
(PART CREATIVE-WRITING PEN)
(PERFORM NORMAN-MAILER CREATIVE-WRITING)
(SUBSTANCE CHAPTER PAPER)
(PART NOVEL CHAPTER)
(PART PLOT SURPRISE)
(PART PLOT EVENT)
(PART PLOT SCENARIO)
(PART PLOT CHARACTER)
(PART NOVEL PLOT)
(PART NOVEL CHARACTER)
(SUBSTANCE NOVEL PAPER)
(AFFECT NORMAN-MAILER NOVEL)
(AFFECT NORMAN-MAILER READER)
(PART VOTER VOTE)
(PART SOCIETY VOTER)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)

```
        (PART           FAMILY-TREE                 FAMILY-RELATIVE)
        (DEPEND         FAMILY-TREE                 FAMILY-BREEDING)
        (DEPEND         FAMILY                      FAMILY-TREE)
        (DEPEND         PERSON                      FAMILY)
        (PART           SOCIETY                     CITIZEN)
        (AFFECT         NORMAN-MAILER               SOCIETY)
        (AFFECT         NORMAN-MAILER               HISTORY))))

SCULPTOR
        (PART           ARTIST-STUDIO               WINDOW)
        (LOCATION-OF    SCULPTOR                    ARTIST-STUDIO)
        (CONTROL        SCULPTOR                    CHISEL)
        (AFFECT         SCULPTOR                    STONE)
        (AFFECT         SCULPTOR                    STATUE)
        (PART           STATUE                      HEAD)
        (PART           STATUE                      LEG)
        (PART           STATUE                      ARM)
        (SUBSTANCE      STATUE                      STONE)
        (CREATE         SCULPTOR                    STATUE))))

HACKER
        (DEPEND         PERSON                      PERSONAL-HEALTH)
        (PART           FAMILY                      FAMILY-RELATIVE)
        (PART           GENE-POOL                   CHARACTERISTIC)
        (PART           GENE-POOL                   GENE)
        (DEPEND         FAMILY                      GENE-POOL)
        (PART           FAMILY-TREE                 FAMILY-RELATIVE)
        (DEPEND         FAMILY-TREE                 FAMILY-BREEDING)
        (DEPEND         FAMILY                      FAMILY-TREE)
        (DEPEND         PERSON                      FAMILY)
        (CONTROL        HACKER                      LOGIC-PROBE)
        (CONTROL        HACKER                      INTERNET)
        (CONTROL        HACKER                      GAME-PROGRAM)
        (CONTROL        HACKER                      COMPUTER)
        (PART           INTERNET                    ETHERNET-CABLE)
        (PART           INTERNET                    COMPUTER-SERVER)
        (AFFECT         HACKER                      INTERNET)
        (CREATE         HACKER                      GAME-PROGRAM)
        (PART           COMPUTER-SERVER             SECURITY)
        (PART           COMPUTER-SERVER             FILE-PARTITION)
        (PURPOSE        COMPUTER-SERVER             STORAGE)
        (AFFECT         HACKING                     COMPUTER-SERVER)
        (PERFORM        HACKER                      HACKING)
        (PART           COMPUTER                    MONITOR)
        (PART           COMPUTER                    CPU)
        (PART           COMPUTER                    KEYBOARD)
        (AFFECT         PROGRAMMING                 COMPUTER)
        (PERFORM        HACKER                      PROGRAMMING)
        (PART           BLUE-JEANS-AND-SNEAKERS     SNEAKERS)
        (PART           BLUE-JEANS-AND-SNEAKERS     BLUE-JEANS)
        (WEAR           HACKER                      BLUE-JEANS-AND-SNEAKERS))))

CRIMINAL
        (CONTROL        CRIMINAL                    GRAPEVINE)
        (PART           OUTLAW-GANG                 HENCHMAN)
        (CONTROL        CRIMINAL                    OUTLAW-GANG)
        (CONTROL        CRIMINAL                    LOCK-PICK)
        (AFFECT         THERMAL-LANCE               WALL-SAFE)
        (CONTROL        CRIMINAL                    THERMAL-LANCE)
        (PART           CRIMINAL                    BLACK-GLOVE)
        (PART           STOCKING-MASK               STOCKING)
        (SUBSTANCE      STOCKING-MASK               NYLON)
        (PART           CRIMINAL                    STOCKING-MASK)
        (PART           POLICE-FORCE                POLICEMAN)
        (AFFECT         CRIMINAL                    POLICE-FORCE)
        (AFFECT         CRIMINAL                    VICTIM)
        (AFFECT         CRIMINAL                    BANK)
        (AFFECT         CRIMINAL                    DOOR-LOCK)
        (PART           WALL-SAFE                   TUMBLER)
        (DEPEND         WALL-SAFE                   SAFE-COMBINATION)
        (AFFECT         CRIMINAL                    WALL-SAFE)
        (PART           EDIFACE                     OCCUPANT)
        (PART           EDIFACE                     BLUEPRINT)
        (PART           EDIFACE                     FLOOR)
        (PART           EDIFACE                     CEMENT)
```

(PART EDIFACE MORTAR)
(PART EDIFACE BEAM)
(PART EDIFACE BRICK)
(PART BANK SECURITY)
(PURPOSE BANK STORAGE)
(AFFECT BANK-RAID BANK)
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(AFFECT BANK-RAID VICTIM)
(PERFORM CRIMINAL BANK-RAID)
(AFFECT LOCK-PICK DOOR-LOCK)
(PART BURGLARY LOCK-PICK)
(PERFORM CRIMINAL BURGLARY))))

ACCOUNTANT
(DEPEND PERSON PERSONAL-HEALTH)
(PART FAMILY FAMILY-RELATIVE)
(PART GENE-POOL CHARACTERISTIC)
(PART GENE-POOL GENE)
(DEPEND FAMILY GENE-POOL)
(PART FAMILY-TREE FAMILY-RELATIVE)
(DEPEND FAMILY-TREE FAMILY-BREEDING)
(DEPEND FAMILY FAMILY-TREE)
(DEPEND PERSON FAMILY)
(PART OFFICE TELEPHONE)
(PART TYPEWRITER CARRIAGE-RETURN)
(PART TYPEWRITER KEYBOARD)
(PART OFFICE TYPEWRITER)
(PART OFFICE DESK)
(LOCATION-OF ACCOUNTANT OFFICE)
(PART ACCOUNTANT POCKET-PROTECTOR)
(WEAR ACCOUNTANT THREE-PIECE-SUIT)
(PART PEN INK)
(CONTROL ACCOUNTANT PEN)
(CONTROL ACCOUNTANT SPREADSHEET)
(AFFECT ACCOUNTANT SPREADSHEET)
(PART LEDGER COLUMN)
(SUBSTANCE LEDGER PAPER)
(CREATE ACCOUNTANT LEDGER)
(PERFORM ACCOUNTANT ACCOUNTANCY))))

MAGICIAN
(EFFECT JUG-TRICK CONVERSION)
(AFFECT JUG-TRICK WATER)
(CREATE JUG-TRICK WINE)
(PERFORM MAGICIAN JUG-TRICK)
(PERFORM MAGICIAN ILLUSION)
(PERFORM MAGICIAN MAGIC)
(DEPEND MAGICIAN AUDIENCE)
(PART AUDIENCE MEMBER-OF-AUDIENCE)
(CONTROL MAGICIAN AUDIENCE)
(PART MAGICIAN FORMAL-GLOVE)
(WEAR MAGICIAN BLACK-TUXEDO)
(LOCATION-OF MAGICIAN NIGHTCLUB)
(CONTROL MAGICIAN MAGIC-WAND)
(CONTROL MAGICIAN STAGE-ASSISTANT)
(PERFORM MERLIN COUNSEL)
(PERFORM MERLIN MAGIC)
(PART ILLUSION TRICK-MIRROR)
(PART ILLUSION STAGE-RABBIT)
(PART ILLUSION STAGE-ASSISTANT)
(PART THEATRE CURTAIN)
(PART THEATRE STAGE)
(LOCATION-OF ILLUSION THEATRE)
(PART MAGIC ILLUSION)
(PART MAGIC FAITH)
(PART MAGIC SUSPENSION-OF-DISBELIEF)
(CONTROL MERLIN MAGIC)

```
(CONTROL        MORGANA-LEFAY   MORDRED)
(CONTROL        KNIGHT          HORSE)
(SUBSTANE       SWORD           STEEL)
(CONTROL        KNIGHT          SWORD)
(UP             MORDRED         MORGANA-LEFAY)
(DOWN           MORGANA-LEFAY   MORDRED)
(CONTROL        THE-EMPEROR     DARTH-VADER)
(MODE           THE-EMPIRE      BELLIGERANT)
(MODE           THE-EMPIRE      EVIL)
(PART           THE-EMPIRE      STORM-TROOPER)
(CONTROL        THE-EMPEROR     THE-EMPIRE)
(CONTROL        THE-EMPEROR     THE-FORCE)
(CONTROL        OBI-WAN-KENOBI  THE-FORCE)
(CONTROL        OBI-WAN-KENOBI  LIGHT-SABER)
(DISCONNECT     OBI-WAN-KENOBI  THE-EMPORER)
(SUBSTANCE      MAN             HUMAN-BONE)
(SUBSTANCE      MAN             HUMAN-FLESH)
(PART           HUMAN-HAIR      HAIR-FOLLICLE)
(PART           MAN             HUMAN-HAIR)
(PART           HUMAN-SKIN      HAIR-FOLLICLE)
(PART           HUMAN-SKIN      PORE)
(PART           MAN             HUMAN-SKIN)
(PART           MAN             HAND)
(PART           EAR             EAR-LOBE)
(LOCATION-OF    EAR             HEAD)
(PART           MAN             EAR)
(PART           MAN             NOSE)
(PART           MAN             EYE)
(PART           MOUTH           TEETH)
(LOCATION-OF    MOUTH           FACE)
(PART           FACE            MOUTH)
(LOCATION-OF    NOSTRIL         NOSE)
(LOCATION-OF    NOSTRIL         FACE)
(PART           NOSE            NOSTRIL)
(LOCATION-OF    NOSE            FACE)
(PART           FACE            NOSE)
(LOCATION-OF    EYE             FACE)
(PART           FACE            EYE)
(LOCATION-OF    FACE            HEAD)
(PART           MAN             FACE)
(CONTROL        DARTH-VADER     DEATH-STAR)
(CONTROL        DARTH-VADER     THE-FORCE)
(CONTROL        DARTH-VADER     LIGHT-SABER)
(PART           DARTH-VADER     MASK)
(PART           DARTH-VADER     CLOAK)
(DISCONNECT     DARTH-VADER     PRINCESS-LEIA)
(DISCONNECT     OBI-WAN-KENOBI  DARTH-VADER)
(DISCONNECT     THE-EMPEROR     OBI-WAN-KENOBI)
(DEPEND         PERSON          PERSONAL-HEALTH)
(PART           FAMILY          FAMILY-RELATIVE)
(PART           GENE-POOL       CHARACTERISTIC)
(PART           GENE-POOL       GENE)
(DEPEND         FAMILY          GENE-POOL)
(PART           FAMILY-TREE     FAMILY-RELATIVE)
(DEPEND         FAMILY-TREE     FAMILY-BREEDING)
(DEPEND         FAMILY          FAMILY-TREE)
(DEPEND         PERSON          FAMILY)
(CONTROL        SPACE-REBEL     X-WING-FIGHTER)
(PART           REBEL-ALLIANCE  SPACE-REBEL)
(MODE           REBEL-ALLIANCE  SYMPATHETIC)
(MODE           REBEL-ALLIANCE  HEROIC)
(MODE           REBEL-ALLIANCE  GOOD)
(CONTROL        PRINCESS-LEIA   REBEL-ALLIANCE)
(DISCONNECT     THE-EMPEROR     PRINCESS-LEIA))))
```

The Sundry domains collection

```
;;; ================================================
;;; ===============Background Knowledge ==============
;;; ================================================

3-BEARS          (taller-than          daddy-bear          mommy-bear)
                 (taller-than          mommy-bear          baby-bear)
                 (taller-than          daddy-bear          baby-bear)))

APPLE            (inside               apple-core          apple)
                 (made-of              apple               veg-substance)
                 (found                apple               tree)
                 (has-part             apple               apple-core)))

ARMY             (surround             fortress            swamp)
                 (avoid                army                swamp)
                 (split-into           army                platoon)
                 (part-of              army                platoon)
                 (go-down              army                road)
                 (enable               split-into          go-down)
                 (attack               platoon             fortress)
                 (conquer              army                fortress)
                 (and                  attack              conquer)))

ARTHURIAN-SAGA   (cause                help                become)
                 (help                 merlin              arthur          obtain)
                 (obtain               arthur              excalibur)
                 (become               arthur              king)))

ASSASINATE-JFK   (lived-in             Jfk                 white-house)
                 (assasinate           Oswald              Jfk)
                 (assasinate           Ruby                Oswald)))

ASSASINATE -pig ; an-Anti-domain
                 (lived-in             pig                 pig-house)))

ATOM             (heavier              nucleus             electron) ; needs inferences
                 (attracts             nucleus             electron)))

ATOM-CLONE ;to calculate distance in structure space
                 (heavier              nucleus             electron)

                 (attracts             nucleus             electron)))

ATOM-Falkenhainer (attracts            nucleus             electron)
                 (heavier              nucleus             electron)       ;needs inferences
                 (attracts             nucleus             electron)
                 (opposite-sign        nucleus             electron)))

BANKER           (hoards               banker              money)
                 (works-in             banker              bank)))

BEAUTIFUL-GAME ; ==== aka              Soccer
                 (result-in            hop                 thud)
                 (hop                  footer              ground)
                 (thud                 footer              wall)))

BIRD
                 (inhabits             bird                sky)
                 (has-part             bird                wings)
                 (flies-through        bird                sky)
                 (enable               has-part            flies-through)))

BURN-PAPER
                 (next-to              candle              paper)
                 (burn                 candle              paper)
                 (cause                next-to             burn)))

BURN-ROCK        ; an anti-domain
                 (next-to              camp-fire           rock)))

BUS              (part-of              bus                 wheel)
                 (part-of              bus                 seat)
                 (contains             bus                 human)
```

	(sit-in	human	bus)
	(transport	bus	human)))
BUY-APPLE	(go-to	john	shop)
	(located-in	apple	shop)
	(buy	john	apple)
	(eat	john	apple)
	(enable	buy	eat)))
CANOEING	(propel	man	cannoe)
	(inside	man	cannoe)
	(paddle	man	cannoe)))
CARAVAGGIO	(control	caravaggio	paintbrush)
	(used-for	paintbrush	painting)
	(influence	caravaggio	italian-school)
	(create	caravaggio	painting)
	(cause	violence	murder)
	(subject-of	painting	real-life)
	(treatment-of	painting	personal)
	(style	painting	theatrical)
	(lifestyle	caravaggio	mobile)
	(style	caravaggio	non-derivitive)
	(died	caravaggio	young)
	(born-in	caravaggio	italy)
	(part-of	paintbrush	bristle)
	(style	painting	moody)))
CHAIR	(made-of	chair	wood)
	(part-of	chair	chair-seat)
	(part-of	chair	chair-back)
	(part-of	chair	chair-leg)
	(connect	chair-back	chair-seat)
	(connect	chair-seat	chair-legs)
	(sit-on	chair	human)))
CLOTHES	(cover-state	clothes	human)
	(decorate	clothes	human)))
COMPOSER-assorted	;from SME		
	(control	composer	orchestra)
	(control	composer	conductor-baton)
	(affect	composer	listenership)
	(part-of	orchestra	musician)
	(part-of	orchestra	percussion)
	(part-of	listenership	listener)
	(part-of	percussion	drum)
	(control	musician	musical-instrument)))
COMPUTER	(control	cpu	computer)
	(execute	cpu	program)))
CREATE-BUILDING	(create	architect	blue-print)
	(examine	builder	blue-print)
	(and	create	examine)
	(build	builder	house)
	(enable	and	build)))
CUT-APPLE	(holds	john	apple)
	(cut	john	apple)
	(enable	hold	cut)))
CAT-BALL	(holds	bob	rain)))
CYCLING	; Analogy of DRIVING		
	(facilitate	on-top-of	propel)
	(propel	man	bike)
	(on-top-of	man	bike)))
CYCLING2			
	(enable	own	on-top-of)
	(own	man	bike)
	(control	man	bike)))
DRIVING	; Analogy of CYCLING		

```
                (facilitate     inside          propel)
                (propel         engine          car)
                (inside         engine          car)
                (drive          man             car)))

DRIVING2        (enable         own             inside)
                (own            man             car)
                (control        man             car)
                (drive          man             car)))

EAGLE           (part-of        eagle           eagle-head)
                (part-of        eagle           eagle-torso)
                (part-of        eagle           wings)
                (made-of        eagle           flesh)
                (connect        eagle-torso     eagle-head)
                (connect        eagle-torso     wings)))

EAT-APPLE       (holds          john            apple)
                (eat            john            apple)
                (enable         hold            eat)))

EAT-BALL        (holds          bob             football)))

FISH            (inhabits       fish            water)
                (has-part       fish            fin)))

FLAT-BALL       ; an Anti-domain
                (hit            flat-ball       wall)
                (cause          bounce          hit)))

FLOWER          (part-of        flower          stem)
                (part-of        flower          flower-bloom)
                (connect        flower          stem)
                (connect        flower          flower-bloom)
                (made-of        flower          veg-substance)))

FORTRESS        (surround       swamp           fortress)
                (attack         army            fortress)
                (part-of        army            platoon)
                (avoid          army            swamp)
                (split-into     army            platoon)
                (enable         split-into      avoid)
                (go-down        platoon         path)
                (converge       platoon         fortress)
                (attack         army            fortress)
                (cause          converge        attack)))

FORTRESS-arcs   ; from ARCS
                (lead_to        roads           fortress)
                (go_down        army            roads)
                (capture        army            obj_fortress)
                (enable         go-down         capture)))

FRUIT           (eat            animal          fruit)
                (inside         fruit           fruit-seed)))

FURNITURE       (support        furniture       human)
                (decorate       furniture       house)
                (used-in        furniture       house)))

GENERAL ; from SME
                (control        general         army)
                (control        general         sword)
                (affect         general         society)
                (part-of        army            soldier)
                (part-of        army            artillery)
                (part-of        society         civilian)
                (part-of        artillery       cannon)
                (control        soldier         weapon)))

GOLF-PLAY       (cause          bounce          hit)
                (bounce         golf-ball       golf-green)
                (hit            golf-ball       flag)))

GUN             (part-of        gun             barrell)
```

```
                (part-of        gun             handle)
                (made-of        gun             metal)
                (shoots-with    human           gun)
                (propel         gun             bullet)
                (damage         bullet          something)))

HAMMER          (connect        hammer-head     hammer-handle)
                (use            human           hammer)
                (damage         hammer          wall)))

HEAT-FLOW       (cause          greater         flow)
                (greater        temperature-a   temperature-b)
                (flow-from      heat            coffee)
                (flow-to        heat            ice-cube)
                (flow-along     heat            iron-bar)))

HEAT-FLOW-good  ; ***********
                (temperature-of coffee          temperature-a)
                (temperature-of iron-bar        temperature-b)
                (flow-from      heat            coffee)
                (flow-to        heat            iron-bar)
                (greater        temperature-a   temperature-b)
                (cause          greater         flow)))

HORSE           (transport      horse           human)
                (made-of        horse           flesh)
                (has-part       horse           horse-head)
                (has-part       horse           horse-torso)
                (has-part       horse           horse-legs)
                (connect        horse-head      horse-torso)
                (connect        horse-legs      horse-torso)))

HOUSE           (made-of        house           brick)
                (live-in        house           human)))

INSECT          (has-part       insect          insect-head)
                (has-part       insect          insect-legs)
                (has-part       insect          insect-body)
                (connect        insect-head     insect-body)
                (connect        insect-head     insect-legs)))

JOHN-DOE-DRIVE ; an anti-domain
                (inside         john-doe        car)
                (enable         inside          drive)))

KENNEDY-SAGA
                (cause          help            become)
                (help           joe-kennedy     jfk     obtain)
                (obtain         jfk             democratic-nomination)
                (become         jfk             president)))

KICK-ABOUT      (kick           tom             football)))

KNIFE-CUT       (has-part       knife           blade)
                (has-part       knife           knife-handle)
                (connect        blade           knife-handle)
                (made-of        knife           wood)
                (made-of        knife-handle    wood)
                (next-to        knife           table)
                (use            human           knife)
                (cut            human           something)
                (damage         knife           something)))

LADA-CAR        (is-a           lada            car)))

LEADBELLY       (plays          leadbelly       guitar)
                (part-of        music           note)
                (has            note            tone)
                (influence      leadbelly       black-musicans)
                (create         leadbelly       music)
                (cause          violence        murder)
                (theme          music           human)
                (treatment-of   music           personal)
                (style          music           graphic)
                (lifestyle      leadbelly       mobile)
```

```
                (died           leadbelly       middle-age)
                (born-in        leadbelly       alabama)
                (part-of        guitar          string)
                (has-part       music           rythm)
                (owns           leadbelly       guitar)))

LOVE-TRIANGLE   (loves          tom             mary)
                (loves          mary            joe)
                (loves          joe             tom)))

MAN-MIND        (control        brain           man)
                (part-of        brain           mind)))

MELT-BRICK   ; An anti-domain
                (see            john            sun)
                (expose         john            brick)))

MELT-SNOW       (see            mary            sun)
                (expose         mary            snow)
                (melt           sun             snow)
                (cause          expose          melt)))

ORANGE          (inside         orange-core     orange)
                (made-of        orange          veg-substance)
                (next-to        orange          tree)
                (has-part       apple           orange-peel)))

RECTANGLE-AREA
                (product        length          breadth)
                (length         rectangle)
                (breadth        rectangle)
                (area           rectangle)
                (proportional   product         area)))

ROLLS-ROYCE-CAR (is-a           rolls-royce     car)
                (expensive      rolls-royce)))

SCISSORS-CUT    (cut            scissors        something)
                (connect        blade1          blade2)
                (part-of        blade1          scissors)
                (part-of        blade2          scissors)))

SEAT-DRIVE      ; AN Anti-domain
                (inside         seat            car)
                (enable         inside          drive)))

SHOE            (has-part       human           foot)
                (inside         shoe            foot)
                (has-part       shoe            shoe-sole)
                (has-part       shoe            shoe-upper)
                (connect        shoe            shoe-sole)
                (connect        shoe            shoe-upper)))

SOCCER          (cause          bounce          hit)
                (bounce         football        field)
                (hit            player          ball)
                (plays          team            soccer)
                (part-of        team            goal-keeper)
                (keep-out       keeper          ball   goal)
                (played-with    soccer          boots)
                (part-of        boots           laces)
                (part-of        clubs           studs)
                (part-of        pitch           net)
                (kick           goal-hanger     ball)
                (enter          ball            net)))

SOLAR-SYSTEM    (heavier        sun             earth)
                (attracts       sun             earth)
                (revolves       earth           sun)
                (and            heavier         attracts)
                (cause          and             revolves)))

SOLAR-SYSTEM-Falkenhainer
                (cause          attracts        revolves)
                (attracts       sun             earth)
```

```
                (heavier        sun             earth)
                (revolves       earth           sun)))

SPIDER          (has-part       spider          spider-legs)
                (has-part       spider          spider-body)
                (connect        spider-legs     spider-body)))

STORY-TELL      (hear           bob             story)
                (tell           bob             story)
                (enable         hear            tell)))

SUN             (heavier        sun             planet)
                (attracts       sun             planet)
                (revolves       planet          sun)
                (and            heavier         attracts)
                (cause          heavier         attracts)
                (enable         oxygen-atmosphere habitation)))

SURGEON-Assorted
                (control        surgeon         scalpel)
                (create         surgeon         operating-procedure)
                (affect         surgeon         sick-people)
                (part-of        sick-people     patient)
                (control        surgeon         medical-staff)
                (part-of        medical-staff   medical-asistants)
                (part-of        medical-assistants)
                (part-of        medical-staff   junior-surgeon)
                (control        medical-assistants instruments)
                (type-of        instruments     scalpel)
                (control        junior-surgeon      crash-cart)))

THROW-BALL      (find           tom             football)
                (throw          tom             football)
                (enable         find            throw)))

THROW-GUN       (find           tom             gun)))

THROW-HOUSE ; An anti-domain
                (find           tom             house)))

TOOL            (repair         human           tool   object)
                (make           human           tool   object)))

TRIANGLE        (line1          p1              p2)
                (line2          p2              p3)
                (line3          p1              p3)))

TRIANGLE-DIRECTED
                (directed-line1 p1              p2)
                (directed-line2 p2              p3)
                (directed-line3 p1              p3)))

TUMOR           (surround       healthy-tissue  tumour)
                (attack         x-ray           tumour)
                (part-of        x-ray           beam)
                (avoid          x-ray           healthy-tissue)
                (split-into     x-ray           beam)
                (enable         split-into      avoid)
                (go-down        beam            path)
                (converge       beam            tumour)
                (attack         x-ray           tumour)
                (cause          converge        attack)                 ))

REQUITED-LOVE   (loves          tom             mary)
                (loves          mary            joe)
                (loves          joe             mary)
                (jealous-of     tom             joe)
                (cause          loves           jealous-of))) ; only tom is unloved

VAMPIRE         (hoards         vampire         blood)
                (lusts-after    vampire         blood)
                (lives-in       vampire         coffin)
                (cause          lives-in        lusts-after)))

VEHICLE         (predicates     (has-part       vehicle         wheel)
```

	(has-part	vehicle	vehicle-body)	
	(contains	vehicle	human)	
	(transport	vehicle	human)))	
VICTIM	(children		john-doe	jane-doe))
WATER-FLOW	(greater-pressure	beaker	vial)	
	(flow-from	water	beaker)	
	(flow-to	water	vial)	
	(cause	greater-pressure	flow)	
	(greater	diameter-a	diameter-b)))	
WEAPON	(injure	weapon	person)))	

Appendix B1

Retrieval results for the Professions domains

Retrieval	Accountant	Architect	Author	Butcher	Chef	Composer	Criminal	General	Hacker	Magician	Politician	Priest	Scientist	Sculptor
Accountant	3	68	187	11	63	80	47	188	15	164	122	125	48	42
Architect	66	3	124	56	10	17	21	124	52	98	56	59	21	109
Author	184	121	3	175	124	107	141	14	171	42	69	67	139	226
Butcher	9	58	177	3	54	70	37	178	6	155	112	116	39	52
Chef	61	10	126	51	3	18	16	127	48	103	60	63	15	104
Composer	77	15	109	68	16	3	33	110	64	86	43	47	31	120
Criminal	44	23	143	35	19	36	3	144	31	119	77	80	7	87
General	185	121	13	176	124	107	141	3	172	34	67	65	139	228
Hacker	13	55	174	4	50	66	33	175	3	151	108	112	35	55
Magician	162	96	44	152	100	84	117	36	149	3	43	41	116	205
Politician	119	53	72	110	57	41	74	70	106	45	3	7	73	162
Priest	123	57	70	113	61	45	78	67	110	43	6	3	76	166
Scientist	46	23	141	36	17	34	6	142	33	118	75	79	3	89
Sculptor	44	111	229	54	106	123	90	230	58	207	165	168	91	3

Mapping results for the Professions domains

Mappings	Accountant	Architect	Author	Butcher	Chef	Composer	Criminal	General	Hacker	Magician	Politician	Priest	Scientist	Sculptor
Accountant	24	25	35	23	23	22	22	21	25	19	23	22	25	5
Architect	21	50	84	24	42	52	40	68	28	32	46	47	40	4
Author	22	47	90	28	47	55	41	57	28	33	48	30	41	5
Butcher	22	24	41	29	24	22	15	24	27	22	19	18	24	4
Chef	24	43	68	28	49	52	41	51	29	34	44	40	43	5
Composer	23	45	73	28	48	56	42	62	28	33	53	42	42	5
Criminal	21	38	55	23	42	43	42	49	28	29	38	37	37	5
General	24	50	90	28	49	56	41	98	29	53	66	63	43	5
Hacker	22	29	49	25	32	27	24	31	30	17	28	25	29	4
Magician	24	50	89	28	49	55	41	90	29	90	71	73	43	5
Politician	23	50	89	27	48	55	42	87	28	53	72	67	42	5
Priest	24	50	89	28	49	55	41	90	29	56	71	73	43	5
Scientist	24	40	52	25	41	41	35	37	29	28	40	32	43	5
Sculptor	2	3	13	2	2	3	7	1	3	4	3	4	3	5

Inferences from the Professions domains

	Accountant	Architect	Author	Butcher	Chef	Composer	Criminal	General	Hacker	Magician	Politician	Priest	Scientist	Sculptor
Accountant	0	3	2	2	0	1	3	0	2	0	1	0	0	0
Architect	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Author	0	2	0	0	0	0	1	0	1	0	0	0	1	0
Butcher	1	3	0	0	0	1	3	0	1	0	1	0	1	0
Chef	0	4	2	1	0	1	4	0	0	0	1	0	1	0
Composer	0	3	1	0	0	0	1	0	2	0	0	0	1	0
Criminal	0	2	1	1	0	0	0	0	1	1	0	1	0	0
General	1	5	2	0	1	1	4	0	2	1	2	1	1	0
Hacker	1	2	2	1	1	1	2	0	0	1	1	1	1	0
Magician	1	7	4	1	2	4	6	4	2	0	4	3	1	1
Politician	0	2	0	1	0	0	1	0	0	0	0	0	0	0
Priest	1	4	1	1	0	0	3	0	1	0	1	0	1	0
Scientist	0	3	2	1	0	1	2	0	1	0	1	0	0	0
Sculptor	0	1	1	2	0	0	2	0	2	0	0	0	0	0

Appendix B2-1

Retrieval from the Assorted domains

Target domains are listed across the top of the following pages. The source domains run vertically down each of these pages.

KNN Distance	3-bears	Apple	Army	Arthurian-saga	Assasinate-JFK	Assasinate-pig	Atom	Atom-clone	atom-falkenhainer	Banker	Beautiful-game	Bird	Burn-paper	Burn-rock	Bus	Buy-apple	Canoeing	Caravaggio	Chair	Clothes
3-Bears	2.6	5.7	18	7.8	3.5	4.6	3	3	5.3	2.8	5.8	6.6	5.9	4.6	7.9	8.2	3.6	30	13	3
Apple	4.6	2.6	14	6.9	5.8	8.1	5.6	5.6	3.5	5.3	6.6	5.7	6.9	8.1	4.5	6.1	3.6	26	9.4	5.6
Army	14	10	2.6	11	16	18	16	16	11	16	14	11	14	18	8.5	8.6	13	16	6.9	16
Arthurian-Saga	4.1	3.9	14	2.6	4.2	6.6	4.4	4.4	4.5	4	2.4	2	3	6.6	6.2	4.1	3.3	27	11	4.4
Assasinate-Jfk	4.9	7.6	19	8.6	2.6	3.5	3.2	3.2	7.1	3.3	5.9	7.4	5.8	3.5	9.9	9.4	4.9	32	15	3.2
Assasinate-Pig	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Atom	4.6	7.4	19	8.7	3.2	3	2.6	2.6	6.9	2.8	6	7.5	5.9	3	9.8	9.4	4.6	32	15	2.6
Atom-Clone	4.6	7.4	19	8.7	3.2	3	2.6	2.6	6.9	2.8	6	7.5	5.9	3	9.8	9.4	4.6	32	15	2.6
Atom-Falkenhainer	4	3.5	15	7.2	5.2	7.5	4.9	4.9	2.6	5	6.6	5.7	6.5	7.5	5	6.3	2.8	27	10	4.9
Banker	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Beautiful-Game	3.7	6	17	5.5	1.7	3.5	2	2	5.9	1.7	2.6	4.6	2.8	3.5	8.5	6.9	3.5	30	14	2
Bird	3.5	3.5	14	4	3.3	6	3.5	3.5	3.6	3.3	3	2.6	3.2	6	5.9	4.5	2	28	11	3.5
Burn-Paper	3.9	6.2	17	5.7	1.4	3.3	1.7	1.7	5.8	2	2.8	4.7	2.6	3.3	8.7	7	3.3	31	14	1.7
Burn-Rock	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Bus	6.2	2.8	13	7.6	7.9	10	7.7	7.7	3.6	7.5	8.3	6.6	8.5	10	2.6	6	5.3	24	7.2	7.7
Buy-Apple	4.8	2.2	12	4.1	5.7	8.4	5.7	5.7	2.8	5.7	4.9	2.8	5	8.4	4	2.6	3.3	25	9.1	5.7
Canoeing	3.6	5	17	7.4	3.5	5.2	3	3	4.5	3.2	5.7	6	5.6	5.2	7.2	7.4	2.6	29	13	3
Caravaggio	28	24	16	26	30	32	30	30	25	29	28	26	29	32	22	23	27	2.6	17	30
Chair	11	7.3	11	11	13	15	13	13	8.4	12	13	10	13	15	5.3	8.9	10	19	2.6	13
Clothes	4.6	7.4	19	8.7	3.2	3	2.6	2.6	6.9	2.8	6	7.5	5.9	3	9.8	9.4	4.6	32	15	2.6
Composer	13	10	11	13	16	18	15	15	11	15	15	13	15	18	8.5	11	13	18	4	15
Computer	5.3	8.1	20	9.1	3.3	2.8	3.2	3.2	7.7	3.3	6.2	7.9	6.2	2.8	11	10	5.3	33	16	3.2
Create-Building	6	5.3	12	2	5.9	8.2	6.3	6.3	5.9	6.1	3.3	2.2	3.7	8.2	7	3.5	5.3	27	11	6.3
Cut-Apple	4.4	5.9	16	5.9	2	4.4	2.6	2.6	5.7	2.8	3.5	4.7	3.3	4.4	8.4	6.7	3.3	30	14	2.6
Cut-Ball	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Cycling	3.9	6.2	17	5.7	1.4	3.3	1.7	1.7	5.8	2	2.8	4.7	2.6	3.3	8.7	7	3.3	31	14	1.7
Cycling2	4.4	5.9	16	5.9	2	4.4	2.6	2.6	5.7	2.8	3.5	4.7	3.3	4.4	8.4	6.7	3.3	30	14	2.6
Driving	3.5	3.5	14	4	3.3	6	3.5	3.5	3.6	3.3	3	2.6	3.2	6	5.9	4.5	2	28	11	3.5
Driving2	4.6	3.9	14	5	4	6.9	4.4	4.4	3.7	4.5	4.2	3.5	4.1	6.9	6	4.6	2.6	27	11	4.4
Eagle	7.4	4.6	12	8.8	9.8	12	9.6	9.6	5.3	9.4	9.9	8	10	12	3.2	6.9	7.3	22	5	9.6
Eat-Apple	4.4	5.9	16	5.9	2	4.4	2.6	2.6	5.7	2.8	3.5	4.7	3.3	4.4	8.4	6.7	3.3	30	14	2.6
Eat-Ball	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Fish	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Flat-Ball	6.9	9.5	20	8.8	3.7	3.3	4.4	4.4	9.1	4.7	5.7	7.9	5.4	3.3	12	10	6.6	34	17	4.4
Flower	5.7	3	13	7.7	7.7	10	7.5	7.5	3.5	7.3	8.2	6.6	8.4	10	2.8	6.2	5.2	24	7.3	7.5
Fortress	16	12	2	13	17	20	17	17	13	17	16	13	16	20	10	10	15	14	7.5	17
Fortress-Arcs	4.5	3.7	14	4.7	4.4	6.9	4.7	4.7	4.6	4.4	4.1	3.6	4.5	6.9	6.1	4.7	3.5	27	11	4.7
Fruit	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Furniture	3.5	4.7	17	7.2	3.6	5.3	3.2	3.2	4.6	3	5.6	5.9	5.7	5.3	7	7.3	2.8	29	12	3.2
General	13	10	11	13	16	18	15	15	11	15	15	13	15	18	8.5	11	13	18	4	15
Golf-Play	3.7	6	17	5.5	1.7	3.5	2	2	5.9	1.7	2.6	4.6	2.8	3.5	8.5	6.9	3.5	30	14	2
Gun	9.2	5.5	11	9.4	11	14	11	11	7	11	11	8.9	11	14	4.1	7.6	8.7	21	4.5	11

Hammer	3.9	5	17	7.3	4.5	5.7	4.1	4.1	5.7	3.5	5.8	6.3	6.2	5.7	7.3	7.7	4.1	29	12	4.1
Heat-Flow	6.5	5.7	14	5.8	5.9	8.4	6.6	6.6	6.7	6.2	5.6	5.2	6	8.4	7.5	5.8	5.8	27	11	6.6
Heat-Flow-Good	6.7	4.1	11	5.7	7.3	10	7.8	7.8	5.1	7.6	6.8	4.9	7	10	4.9	3.9	5.7	24	8.5	7.8
Horse	10	6.9	10	11	13	15	12	12	7.8	12	12	10	13	15	4.8	8.5	10	19	2.8	12
House	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Insect	4.7	3.5	14	7.6	7.1	9.4	6.9	6.9	3.6	6.7	7.8	6.6	8	9.4	3.9	6.3	4.9	25	7.7	6.9
John-Doe-Drive	5.7	8.4	19	7.8	2.8	2.6	3.3	3.3	8.1	3.5	4.7	6.9	4.6	2.6	11	9.2	5.6	33	16	3.3
Kennedy-Saga	4.1	3.9	14	2.6	4.2	6.6	4.4	4.4	4.5	4	2.4	2	3	6.6	6.2	4.1	3.3	27	11	4.4
Kick-About	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Knife-Cut	16	12	9.4	15	18	21	18	18	13	18	17	15	18	21	9.9	13	15	14	5.4	18
Lada-Car	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Leadbelly	28	24	16	26	30	32	30	30	25	30	29	26	29	32	22	24	27	5.2	18	30
Love-Triangle	2.6	5.7	18	7.8	3.5	4.6	3	3	5.3	2.8	5.8	6.6	5.9	4.6	7.9	8.2	3.6	30	13	3
Man-Mind	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Melt-Brick	4.5	7.2	19	8.5	3.3	3.2	2.8	2.8	7	2.6	5.9	7.4	6	3.2	9.6	9.4	4.7	32	15	2.8
Melt-Snow	3.3	3.9	14	4.1	3.2	5.7	3.3	3.3	4	3.2	2.8	2.8	3	5.7	6.3	4.6	2.2	28	11	3.3
Rectangle-Area	9.7	11	18	8.8	7.1	8.2	8.2	8.2	11	8.5	6.9	8.4	6.6	8.2	13	10	9.1	33	18	8.2
Rolls-Royce-Car	5.3	8.1	20	9.1	3.3	2.8	3.2	3.2	7.7	3.3	6.2	7.9	6.2	2.8	11	10	5.3	33	16	3.2
Scissors-Cut	3.9	3.6	15	7	5.7	7.5	5.4	5.4	4.5	4.9	6.5	6	6.9	7.5	5.5	6.6	4.1	27	9.6	5.4
Seat-Drive	5.7	8.4	19	7.8	2.8	2.6	3.3	3.3	8.1	3.5	4.7	6.9	4.6	2.6	11	9.2	5.6	33	16	3.3
Shoe	7.7	4.7	12	8.9	10	12	9.9	9.9	5.4	9.6	10	8.2	10	12	3	7.1	7.5	22	4.9	9.9
Soccer	24	20	13	22	26	28	26	26	21	25	24	22	25	28	18	20	23	8.7	13	26
Solar-System	6.1	5.4	12	2.6	5.7	8.2	6.1	6.1	5.3	6.2	3.5	2	3.3	8.2	6.9	3.3	4.8	27	12	6.1
Solar-System-Falkenhainer	3.6	3.9	14	4.4	3.2	5.9	3.3	3.3	3.5	3.5	3.2	2.8	3	5.9	6.2	4.6	1.7	28	12	3.3
Spider	2.8	5.3	17	7.5	3.3	4.7	2.8	2.8	5	2.6	5.6	6.2	5.7	4.7	7.5	7.7	3.2	30	12	2.8
Story-Tell	3.9	6.2	17	5.7	1.4	3.3	1.7	1.7	5.8	2	2.8	4.7	2.6	3.3	8.7	7	3.3	31	14	1.7
Sun	9	6.7	9.6	5.6	8.9	12	9.6	9.6	7.2	9.6	7.3	5.3	7.4	12	7.1	4.1	7.8	24	10	9.6
Surgeon	20	17	13	19	22	25	22	22	18	22	22	19	22	25	15	17	20	12	9.2	22
Throw-Ball	3.9	6.2	17	5.7	1.4	3.3	1.7	1.7	5.8	2	2.8	4.7	2.6	3.3	8.7	7	3.3	31	14	1.7
Throw-Gun	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Throw-House	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6
Tool	3.5	6	18	8	3.9	4.2	2.8	2.8	5.7	2.6	5.9	6.9	6	4.2	8.3	8.5	3.7	30	13	2.8
Triangle	4.8	4.4	15	7.7	6.9	8.2	6.2	6.2	5.5	5.7	7.3	6.9	7.8	8.2	6	7.4	5.2	27	9.6	6.2
Triangle-Directed	4.8	4.4	15	7.7	6.9	8.2	6.2	6.2	5.5	5.7	7.3	6.9	7.8	8.2	6	7.4	5.2	27	9.6	6.2
Tumor	16	12	2	13	17	20	17	17	13	17	16	13	16	20	10	10	15	14	7.5	17
Requited-Love	3.3	3	13	3	4.9	7.1	4.6	4.6	2.8	4.5	3.5	2	3.6	7.1	4.9	3.3	2.6	27	9.9	4.6
Vampire	3.5	3.5	14	4	3.3	6	3.5	3.5	3.6	3.3	3	2.6	3.2	6	5.9	4.5	2	28	11	3.5
Water-Flow	5.8	4.5	13	5.3	5.4	8.2	6	6	5	5.9	5.2	4.1	5.3	8.2	6.2	4.5	4.5	26	11	6
Weapon	6.7	10	22	11	4.9	2.6	4.6	4.6	9.6	4.7	7.5	9.6	7.4	2.6	12	12	7.1	35	18	4.6

KNN Distance	Ccompser	Computer	Create-building	cut-apple	cut-ball	Cycling	Cycling2	Driving	Driving2	Eagle	Eat-apple	Eat-ball	Fish	Flat-ball	Flower	Fortress	Fortress-arcs	Fruit	Furniture	General
3-Bears	15	3.5	10	5.9	4.6	5.9	5.9	6.6	7	9.5	5.9	4.6	2.8	6.9	7.5	19	6.9	2.8	3.5	15
Apple	12	6.2	8.9	6.2	8.1	6.9	6.2	5.7	5.6	6.7	6.2	8.1	5.3	8.9	4.6	16	5.5	5.3	3.2	12
Army	8.9	16	10	13	18	14	13	11	11	8	13	18	16	17	9.2	4.5	10	16	13	8.9
Arthurian-Saga	14	4.7	4.9	2.6	6.6	3	2.6	2	3	8.3	2.6	6.6	4	5.7	6.2	16	2.4	4	2.8	14
Assasinate-Jfk	18	2.6	11	5.7	3.5	5.8	5.7	7.4	7.5	12	5.7	3.5	3.3	5.1	9.8	21	7.7	3.3	5	18
Assasinate-Pig	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Atom	18	2.4	11	5.9	3	5.9	5.9	7.5	7.7	12	5.9	3	2.8	5.6	9.6	21	7.9	2.8	4.7	18
Atom-Clone	18	2.4	11	5.9	3	5.9	5.9	7.5	7.7	12	5.9	3	2.8	5.6	9.6	21	7.9	2.8	4.7	18
Atom-Falkenhainer	13	5.6	9.3	6	7.5	6.5	6	5.7	5.5	7.2	6	7.5	5	8.4	4.9	17	6.1	5	3	13
Banker	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Beautiful-Game	16	1.7	7.4	2.8	3.5	2.8	2.8	4.6	5.1	11	2.8	3.5	1.7	3.5	8.5	19	5	1.7	3.3	16
Bird	14	3.9	6.1	2.4	6	3.2	2.4	2.6	2.8	8.2	2.4	6	3.3	5.5	6	16	3	3.3	1.7	14
Burn-Paper	17	1.4	7.6	2.6	3.3	2.6	2.6	4.7	5	11	2.6	3.3	2	3	8.7	19	5.3	2	3.5	17
Burn-Rock	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Bus	10	8.4	9.4	7.9	10	8.5	7.9	6.6	6.3	4.7	7.9	10	7.5	11	2.8	14	6.4	7.5	5	10
Buy-Apple	12	6.3	5.7	4.1	8.4	5	4.1	2.8	2.2	6.2	4.1	8.4	5.7	7.7	4.4	14	2.4	5.7	3.2	12
Canoeing	15	3.5	9.6	5.2	5.2	5.6	5.2	6	5.9	9.4	5.2	5.2	3.2	6.6	7.1	19	6.3	3.2	2.8	15
Caravaggio	16	30	25	28	32	29	28	26	25	20	28	32	29	32	22	14	25	29	27	16
Chair	5.3	14	12	12	15	13	12	10	10	3.6	12	15	12	16	5.4	12	9.9	12	10	5.3
Clothes	18	2.4	11	5.9	3	5.9	5.9	7.5	7.7	12	5.9	3	2.8	5.6	9.6	21	7.9	2.8	4.7	18
Composer	2.6	16	14	15	18	15	15	13	13	6	15	18	15	18	8.5	12	12	15	13	2.6
Computer	18	2.6	11	6.2	2.8	6.2	6.2	7.9	8.1	13	6.2	2.8	3.3	5.3	10	22	8.3	3.3	5.4	18
Create-Building	14	6.6	2.6	3.2	8.2	3.7	3.2	2.2	2.8	8.7	3.2	8.2	6.1	6.3	7.2	14	1.7	6.1	5	14
Cut-Apple	16	2.4	7.6	2.6	4.4	3.3	2.6	4.7	4.6	11	2.6	4.4	2.8	3.6	8.4	18	4.9	2.8	3.5	16
Cut-Ball	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Cycling	17	1.4	7.6	2.6	3.3	2.6	2.6	4.7	5	11	2.6	3.3	2	3	8.7	19	5.3	2	3.5	17
Cycling2	16	2.4	7.6	2.6	4.4	3.3	2.6	4.7	4.6	11	2.6	4.4	2.8	3.6	8.4	18	4.9	2.8	3.5	16
Driving	14	3.9	6.1	2.4	6	3.2	2.4	2.6	2.8	8.2	2.4	6	3.3	5.5	6	16	3	3.3	1.7	14
Driving2	14	4.7	6.6	3	6.9	4.1	3	3.5	2.6	8.4	3	6.9	4.5	5.9	6.2	16	3.5	4.5	2.8	14
Eagle	7.7	10	10	9.6	12	10	9.6	8	7.9	2.6	9.6	12	9.4	13	3	13	7.6	9.4	6.9	7.7
Eat-Apple	16	2.4	7.6	2.6	4.4	3.3	2.6	4.7	4.6	11	2.6	4.4	2.8	3.6	8.4	18	4.9	2.8	3.5	16
Eat-Ball	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Fish	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Flat-Ball	20	3.5	10	5.4	3.3	5.4	5.4	7.9	7.9	14	5.4	3.3	4.7	2.6	12	22	8.2	4.7	6.8	20
Flower	10	8.2	9.6	7.9	10	8.4	7.9	6.6	6.6	4.6	7.9	10	7.3	11	2.6	15	6.6	7.3	4.9	10
Fortress	8.8	18	12	15	20	16	15	13	12	9.1	15	20	17	19	11	2.6	12	17	15	8.8
Fortress-Arcs	14	5	6.2	3.5	6.9	4.5	3.5	3.6	3.5	8.1	3.5	6.9	4.4	6.3	6.3	16	2.6	4.4	3	14
Fruit	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Furniture	15	3.6	9.4	5.3	5.3	5.7	5.3	5.9	6	9.2	5.3	5.3	3	6.8	6.9	19	6.1	3	2.6	15
General	2.6	16	14	15	18	15	15	13	13	6	15	18	15	18	8.5	12	12	15	13	2.6
Golf-Play	16	1.7	7.4	2.8	3.5	2.8	2.8	4.6	5.1	11	2.8	3.5	1.7	3.5	8.5	19	5	1.7	3.3	16
Gun	7.4	12	11	11	14	11	11	8.9	8.8	4	11	14	11	14	4.5	13	8.2	11	8.2	7.4

Hammer	14	4.5	9.4	5.9	5.7	6.2	5.9	6.3	6.7	9.1	5.9	5.7	3.5	7.4	7.3	19	6	3.5	3.5	14
Heat-Flow	14	6.7	6.6	4.9	8.4	6	4.9	5.2	4.9	9.2	4.9	8.4	6.2	7.2	7.9	16	3.6	6.2	5.4	14
Heat-Flow-Good	11	8.2	6.3	5.9	10	7	5.9	4.9	4.1	6.2	5.9	10	7.6	9.2	5.4	13	3.5	7.6	5.5	11
Horse	5.7	13	12	12	15	13	12	10	10	3.2	12	15	12	15	4.9	12	9.6	12	9.6	5.7
House	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Insect	10	7.7	9.4	7.6	9.4	8	7.6	6.6	6.6	4.7	7.6	9.4	6.7	10	3.5	15	6.4	6.7	4.6	10
John-Doe-Drive	19	2.4	9.4	4.6	2.6	4.6	4.6	6.9	7.1	13	4.6	2.6	3.5	2.6	11	21	7.2	3.5	5.7	19
Kennedy-Saga	14	4.7	4.9	2.6	6.6	3	2.6	2	3	8.3	2.6	6.6	4	5.7	6.2	16	2.4	4	2.8	14
Kick-About	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Knife-Cut	6	19	16	17	21	18	17	15	15	8.2	17	21	18	21	10	9.6	14	18	15	6
Lada-Car	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Leadbelly	17	30	25	28	32	29	28	26	25	21	28	32	30	31	23	15	25	30	27	17
Love-Triangle	15	3.5	10	5.9	4.6	5.9	5.9	6.6	7	9.5	5.9	4.6	2.8	6.9	7.5	19	6.9	2.8	3.5	15
Man-Mind	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Melt-Brick	17	2.6	11	6	3.2	6	6	7.4	7.7	12	6	3.2	2.6	5.8	9.5	21	7.7	2.6	4.6	17
Melt-Snow	14	3.7	6	2.2	5.7	3	2.2	2.8	3	8.4	2.2	5.7	3.2	5.2	6.4	16	2.8	3.2	2	14
Rectangle-Area	20	7.6	8.8	6.2	8.2	6.6	6.2	8.4	8.1	15	6.2	8.2	8.5	4.6	13	20	8.5	8.5	9.4	20
Rolls-Royce-Car	18	2.6	11	6.2	2.8	6.2	6.2	7.9	8.1	13	6.2	2.8	3.3	5.3	10	22	8.3	3.3	5.4	18
Scissors-Cut	12	6	8.9	6.4	7.5	6.9	6.4	6	6.2	6.9	6.4	7.5	4.9	8.8	5.4	17	5.5	4.9	3.5	12
Seat-Drive	19	2.4	9.4	4.6	2.6	4.6	4.6	6.9	7.1	13	4.6	2.6	3.5	2.6	11	21	7.2	3.5	5.7	19
Shoe	7.9	11	10	9.9	12	10	9.9	8.2	8.1	2.8	9.9	12	9.6	13	2.8	13	7.9	9.6	7.1	7.9
Soccer	11	26	21	24	28	25	24	22	22	16	24	28	25	28	19	13	21	25	23	11
Solar-System	15	6.3	3.5	2.6	8.2	3.3	2.6	2	1.7	9	2.6	8.2	6.2	5.9	7.1	14	2.8	6.2	4.9	15
Solar-System-Falkenhainer	15	3.7	6.3	2.2	5.9	3	2.2	2.8	2.6	8.5	2.2	5.9	3.5	5.2	6.2	16	3.5	3.5	2	15
Spider	15	3.3	9.6	5.5	4.7	5.7	5.5	6.2	6.5	9.4	5.5	4.7	2.6	6.6	7.3	19	6.4	2.6	3	15
Story-Tell	17	1.4	7.6	2.6	3.3	2.6	2.6	4.7	5	11	2.6	3.3	2	3	8.7	19	5.3	2	3.5	17
Sun	13	9.9	4.7	6.4	12	7.4	6.4	5.3	4.4	8.4	6.4	12	9.6	9.5	7.5	12	4.5	9.6	7.7	13
Surgeon	6.6	23	19	21	25	22	21	19	19	12	21	25	22	25	15	12	19	22	20	6.6
Throw-Ball	17	1.4	7.6	2.6	3.3	2.6	2.6	4.7	5	11	2.6	3.3	2	3	8.7	19	5.3	2	3.5	17
Throw-Gun	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Throw-House	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20
Tool	16	3.3	10	6	4.2	6	6	6.9	7.2	10	6	4.2	2.6	6.8	8.1	20	7.1	2.6	3.6	16
Triangle	12	6.9	9.7	7.5	8.2	7.8	7.5	6.9	7.4	7.1	7.5	8.2	5.7	9.8	5.9	17	6.5	5.7	4.5	12
Triangle-Directed	12	6.9	9.7	7.5	8.2	7.8	7.5	6.9	7.4	7.1	7.5	8.2	5.7	9.8	5.9	17	6.5	5.7	4.5	12
Tumor	8.8	18	12	15	20	16	15	13	12	9.1	15	20	17	19	11	2.6	12	17	15	8.8
Requited-Love	13	5.3	5.3	3.6	7.1	3.6	3.6	2	3	6.9	3.6	7.1	4.5	6.9	4.8	15	3.2	4.5	2.4	13
Vampire	14	3.9	6.1	2.4	6	3.2	2.4	2.6	2.8	8.2	2.4	6	3.3	5.5	6	16	3	3.3	1.7	14
Water-Flow	14	6.2	6.2	4	8.2	5.3	4	4.1	3.2	8.2	4	8.2	5.9	6.9	6.6	15	3	5.9	4.4	14
Weapon	20	4	12	7.7	2.6	7.4	7.7	9.6	9.9	14	7.7	2.6	4.7	5.9	12	24	10	4.7	7.2	20

KNN Distance

	Golf-play	Gun	Hammer	Heat-flow	Heat-flow-good	Horse	House	Insect	John-doe-drive	Kennedy-saga	Kick-about	Knife-cut	Lada-car	Leadbelly	Love-triangle	Man-mind	Melt-brick	Melt-snow	Rectangle-area	Rolls-Royce-car
3-Bears	6	11	4	8	9	12	3	7	6	8	5	18	5	30	3	3	3	7	11	3
Apple	7	7	4	7	7	9	5	5	8	7	8	14	8	26	5	5	5	6	12	6
Army	14	7	13	10	7	7	16	10	16	11	18	7	18	16	14	16	16	11	16	16
Arthurian-Saga	2	9	3	4	6	11	4	6	5	3	7	16	7	27	4	4	4	2	8	5
Assasinate-Jfk	6	13	6	9	10	15	3	9	5	9	3	20	3	32	5	3	3	7	10	3
Assasinate-Pig	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Atom	6	13	5	9	11	15	3	9	5	9	3	20	3	32	5	3	3	7	11	2
Atom-Clone	6	13	5	9	11	15	3	9	5	9	3	20	3	32	5	3	3	7	11	2
Atom-Falkenhainer	7	9	4	8	8	10	5	5	8	7	7	15	7	27	4	5	5	6	12	6
Banker	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Beautiful-Game	3	12	4	6	8	13	2	8	2	5	3	19	3	30	4	2	2	4	8	2
Bird	3	9	3	5	6	11	3	6	4	4	6	16	6	28	3	3	3	3	9	4
Burn-Paper	3	12	4	7	9	14	2	8	2	6	3	19	3	31	4	2	2	5	8	1
Burn-Rock	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Bus	8	5	5	8	7	7	8	4	10	8	10	12	10	24	6	8	8	7	14	8
Buy-Apple	5	7	4	4	4	9	6	4	7	4	8	14	8	25	5	6	6	3	10	6
Canoeing	6	11	4	8	9	12	3	7	6	7	5	18	5	29	4	3	3	6	11	3
Caravaggio	28	18	26	25	22	17	29	23	31	26	32	11	32	4	28	29	29	26	31	30
Chair	13	3	10	11	8	3	12	6	15	11	15	7	15	20	11	12	12	11	18	14
Clothes	6	13	5	9	11	15	3	9	5	9	3	20	3	32	5	3	3	7	11	2
Composer	15	6	12	13	10	5	15	8	17	13	18	7	18	19	13	15	15	13	20	16
Computer	6	14	6	9	11	15	3	10	5	9	3	21	3	33	5	3	3	8	10	3
Create-Building	3	9	5	3	4	11	6	7	5	2	8	16	8	26	6	6	6	2	7	7
Cut-Apple	3	12	4	6	8	13	3	8	3	6	4	19	4	30	4	3	3	5	8	2
Cut-Ball	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Cycling	3	12	4	7	9	14	2	8	2	6	3	19	3	31	4	2	2	5	8	1
Cycling2	3	12	4	6	8	13	3	8	3	6	4	19	4	30	4	3	3	5	8	2
Driving	3	9	3	5	6	11	3	6	4	4	6	16	6	28	3	3	3	3	9	4
Driving2	4	9	4	5	6	11	4	6	5	5	7	16	7	27	5	4	4	4	9	5
Eagle	10	4	7	9	7	5	9	3	12	9	12	10	12	23	7	9	9	8	15	10
Eat-Apple	3	12	4	6	8	13	3	8	3	6	4	19	4	30	4	3	3	5	8	2
Eat-Ball	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Fish	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Flat-Ball	6	15	7	9	11	17	5	11	3	9	3	22	3	33	7	5	5	8	8	3
Flower	8	6	5	8	7	7	7	3	10	8	10	12	10	25	6	7	7	7	14	8
Fortress	16	8	15	12	9	7	17	11	18	13	20	6	20	15	16	17	17	13	18	18
Fortress-Arcs	4	9	3	4	5	11	4	6	5	5	7	16	7	27	4	4	4	3	9	5
Fruit	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Furniture	6	10	3	8	9	12	3	7	6	7	5	17	5	29	3	3	3	6	11	4
General	15	6	12	13	10	5	15	8	17	13	18	7	18	19	13	15	15	13	20	16
Golf-Play	3	12	4	6	8	13	2	8	2	5	3	19	3	30	4	2	2	4	8	2
Gun	11	3	8	9	7	5	11	5	13	9	14	9	14	21	9	11	11	9	16	12
Hammer	6	10	3	7	9	12	3	7	6	7	6	17	6	29	4	3	3	6	12	4
Heat-Flow	6	9	5	3	5	11	6	8	6	6	8	16	8	26	6	6	6	5	9	7

Heat-Flow-Good	7	7	5	3	3	8	8	5	8	6	10	13	10	23	7	8	8	5	10	8
Horse	12	3	9	10	8	3	12	6	14	11	15	7	15	20	10	12	12	10	17	13
House	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Insect	8	6	5	8	7	7	7	3	9	8	9	13	9	25	5	7	7	7	13	8
John-Doe-Drive	5	14	6	8	10	16	3	10	3	8	3	21	3	32	6	3	3	7	8	2
Kennedy-Saga	2	9	3	4	6	11	4	6	5	3	7	16	7	27	4	4	4	2	8	5
Kick-About	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Knife-Cut	17	7	15	15	12	5	18	11	20	15	21	3	21	15	16	18	18	15	22	19
Lada-Car	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Leadbelly	29	19	27	24	22	18	30	23	30	26	32	13	32	3	28	30	30	26	31	30
Love-Triangle	6	11	4	8	9	12	3	7	6	8	5	18	5	30	3	3	3	7	11	3
Man-Mind	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Melt-Brick	6	13	5	9	11	14	3	9	5	8	3	20	3	32	4	3	3	7	11	3
Melt-Snow	3	9	3	4	6	11	3	6	4	4	6	16	6	28	3	3	3	3	8	4
Rectangle-Area	7	16	10	8	11	18	8	13	5	9	8	22	8	32	10	8	8	8	3	8
Rolls-Royce-Car	6	14	6	9	11	15	3	10	5	9	3	21	3	33	5	3	3	8	10	3
Scissors-Cut	6	8	3	7	7	9	5	5	8	7	8	15	8	27	4	5	5	6	12	6
Seat-Drive	5	14	6	8	10	16	3	10	3	8	3	21	3	32	6	3	3	7	8	2
Shoe	10	4	7	9	7	5	10	4	12	9	12	10	12	23	8	10	10	8	15	11
Soccer	24	14	22	20	18	14	25	19	27	22	28	9	28	9	24	25	25	22	28	26
Solar-System	3	10	6	4	5	11	6	7	5	3	8	16	8	27	6	6	6	2	7	6
Solar-System-Falkenhainer	3	10	4	5	6	11	3	6	4	4	6	17	6	28	4	3	3	3	8	4
Spider	6	11	3	8	9	12	3	7	6	7	5	18	5	30	3	3	3	6	11	3
Story-Tell	3	12	4	7	9	14	2	8	2	6	3	19	3	31	4	2	2	5	8	1
Sun	7	9	8	4	3	10	10	8	9	6	12	14	12	23	9	10	10	5	9	10
Surgeon	22	11	19	19	16	10	22	15	24	19	25	6	25	13	20	22	22	19	26	23
Throw-Ball	3	12	4	7	9	14	2	8	2	6	3	19	3	31	4	2	2	5	8	1
Throw-Gun	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Throw-House	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4
Tool	6	11	4	9	10	13	3	8	6	8	4	19	4	31	3	3	3	7	12	3
Triangle	7	8	3	8	8	10	6	5	9	8	8	15	8	27	5	6	6	7	14	7
Triangle-Directed	7	8	3	8	8	10	6	5	9	8	8	15	8	27	5	6	6	7	14	7
Tumor	16	8	15	12	9	7	17	11	18	13	20	6	20	15	16	17	17	13	18	18
Requited-Love	3	8	3	5	6	10	4	4	6	3	7	15	7	27	3	4	4	2	9	5
Vampire	3	9	3	5	6	11	3	6	4	4	6	16	6	28	3	3	3	3	9	4
Water-Flow	5	9	5	3	4	11	6	7	6	5	8	15	8	26	6	6	6	4	8	6
Weapon	7	16	8	11	13	17	5	12	6	11	3	23	3	35	7	5	5	9	11	4

KNN Distance	Scissors-cut	Seat-drive	Shoe	Soccer	Solar-system	Solar-system-[illegible]	Spier	Story-tell	Sun	Surgeon	Throw-ball	Throw-gun	Throw-house	Tool	Triangle	triangle-directed	Tumor	Requited-love	vampire	Water-flow	Weapon
3-Bears	5	6	10	26	10	7	3	6	13	13	6	5	5	3	6	8	19	8	7	8	5
Apple	4	8	7	23	9	6	4	7	10	10	7	8	8	4	4	11	16	7	6	6	8
Army	11	16	8	14	10	11	14	14	7	7	14	18	18	14	12	21	4	10	11	10	18
Arthurian-Saga	4	5	8	23	5	3	3	3	8	8	3	7	7	4	5	10	16	4	2	4	7
Assasinate-Jfk	7	5	12	28	10	7	5	6	13	13	6	3	3	5	8	6	21	9	7	9	3
Assasinate-Pig	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Atom	7	5	12	28	11	7	4	6	13	13	6	3	3	4	8	6	21	9	7	9	3
Atom-Clone	7	5	12	28	11	7	4	6	13	13	6	3	3	4	8	6	21	9	7	9	3
Atom-Falkenhainer	4	8	7	24	9	6	4	6	11	11	6	7	7	4	5	10	17	7	6	7	7
Banker	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Beautiful-Game	6	2	11	26	7	5	3	3	10	10	3	3	3	3	7	7	19	7	5	6	3
Bird	4	4	8	24	6	3	3	3	8	8	3	6	6	3	5	9	16	5	3	4	6
Burn-Paper	6	2	11	27	7	5	3	3	11	11	3	3	3	3	7	6	19	7	5	6	3
Burn-Rock	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Bus	4	10	5	21	9	7	6	8	10	10	8	10	10	6	5	14	14	7	7	7	10
Buy-Apple	3	7	6	22	6	3	4	5	7	7	5	8	8	5	5	12	14	4	3	3	8
Canoeing	5	6	10	26	9	6	3	6	12	12	6	5	5	3	6	8	19	8	6	7	5
Caravaggio	24	31	20	8	26	26	27	29	22	22	29	32	32	28	24	36	14	25	26	24	32
Chair	8	15	3	16	12	11	10	13	12	12	13	15	15	11	8	19	12	10	10	10	15
Clothes	7	5	12	28	11	7	4	6	13	13	6	3	3	4	8	6	21	9	7	9	3
Composer	10	17	6	13	15	14	13	15	13	13	15	18	18	14	10	21	12	12	13	13	18
Computer	8	5	13	29	11	8	5	6	14	14	6	3	3	5	9	5	22	10	8	9	3
Create-Building	5	5	9	23	3	3	5	4	6	6	4	8	8	6	6	11	14	4	2	3	8
Cut-Apple	6	3	11	26	7	5	4	3	10	10	3	4	4	4	7	7	18	7	5	6	4
Cut-Ball	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Cycling	6	2	11	27	7	5	3	3	11	11	3	3	3	3	7	6	19	7	5	6	3
Cycling2	6	3	11	26	7	5	4	3	10	10	3	4	4	4	7	7	18	7	5	6	4
Driving	4	4	8	24	6	3	3	3	8	8	3	6	6	3	5	9	16	5	3	4	6
Driving2	5	5	9	24	6	3	4	4	8	8	4	7	7	4	6	10	16	6	3	4	7
Eagle	5	12	3	19	10	8	7	10	11	11	10	12	12	8	5	15	13	8	8	8	12
Eat-Apple	6	3	11	26	7	5	4	3	10	10	3	4	4	4	7	7	18	7	5	6	4
Eat-Ball	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Fish	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Flat-Ball	9	3	14	30	10	8	7	5	13	13	5	3	3	6	10	4	22	10	8	9	3
Flower	4	10	4	21	10	7	5	8	10	10	8	10	10	6	5	13	15	7	7	7	10
Fortress	13	18	9	13	12	13	15	16	9	9	16	20	20	16	13	23	3	12	13	11	20
Fortress-Arcs	4	5	8	23	7	4	4	4	8	8	4	7	7	4	5	10	16	6	4	4	7
Fruit	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Furniture	5	6	9	25	9	6	3	6	12	12	6	5	5	3	6	8	19	8	6	7	5
General	10	17	6	13	15	14	13	15	13	13	15	18	18	14	10	21	12	12	13	13	18
Golf-Play	6	2	11	26	7	5	3	3	10	10	3	3	3	3	7	7	19	7	5	6	3
Gun	6	13	4	17	11	9	9	11	11	11	11	14	14	9	6	17	13	9	9	9	14

Hammer	4	6	9	25	10	7	3	6	12	12	6	6	6	3	5	9	19	8	6	7	6
Heat-Flow	5	6	10	23	7	6	6	6	8	8	6	8	8	7	7	11	16	7	5	4	8
Heat-Flow-Good	4	8	7	20	7	5	6	7	6	6	7	10	10	7	6	13	13	6	5	3	10
Horse	7	14	3	16	12	10	10	13	11	11	13	15	15	11	7	18	12	10	10	10	15
House	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Insect	3	9	5	21	10	7	5	8	11	11	8	9	9	5	4	12	15	7	7	7	9
John-Doe-Drive	8	3	13	29	9	7	5	5	12	12	5	3	3	5	9	5	21	9	7	8	3
Kennedy-Saga	4	5	8	23	5	3	3	3	8	8	3	7	7	4	5	10	16	4	2	4	7
Kick-About	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Knife-Cut	13	20	8	12	16	15	16	18	14	14	18	21	21	16	13	24	10	14	15	14	21
Lada-Car	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Leadbelly	25	30	21	10	26	26	28	29	22	22	29	32	32	28	25	36	15	26	26	24	32
Love-Triangle	5	6	10	26	10	7	3	6	13	13	6	5	5	3	6	8	19	8	7	8	5
Man-Mind	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Melt-Brick	7	5	12	28	11	7	4	6	13	13	6	3	3	4	7	6	21	9	7	9	3
Melt-Snow	4	4	9	24	6	3	2	3	9	9	3	6	6	3	5	9	16	5	3	4	6
Rectangle-Area	11	5	15	29	8	8	9	7	11	11	7	8	8	10	13	9	20	10	8	8	8
Rolls-Royce-Car	8	5	13	29	11	8	5	6	14	14	6	3	3	5	9	5	22	10	8	9	3
Scissors-Cut	3	8	7	23	9	6	3	7	11	11	7	8	8	4	3	11	17	7	6	7	8
Seat-Drive	8	3	13	29	9	7	5	5	12	12	5	3	3	5	9	5	21	9	7	8	3
Shoe	5	12	3	19	11	8	8	10	11	11	10	12	12	8	6	16	13	8	8	8	12
Soccer	20	27	16	3	22	23	23	25	19	19	25	28	28	24	20	31	13	21	22	21	28
Solar-System	6	5	9	24	3	2	5	3	6	6	3	8	8	6	7	11	14	4	2	3	8
Solar-System-Falkenhainer	5	4	9	25	6	3	3	3	9	9	3	6	6	3	6	9	16	5	3	4	6
Spider	5	6	10	26	10	6	3	6	12	12	6	5	5	3	6	8	19	8	6	8	5
Story-Tell	6	2	11	27	7	5	3	3	11	11	3	3	3	3	7	6	19	7	5	6	3
Sun	7	9	9	21	5	5	8	7	3	3	7	12	12	9	9	15	12	6	5	3	12
Surgeon	17	24	12	8	20	20	20	22	18	18	22	25	25	21	16	28	12	18	19	18	25
Throw-Ball	6	2	11	27	7	5	3	3	11	11	3	3	3	3	7	6	19	7	5	6	3
Throw-Gun	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Throw-House	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3
Tool	6	6	10	26	10	7	3	6	13	13	6	4	4	3	6	7	20	8	7	8	4
Triangle	3	9	7	22	10	7	4	8	12	12	8	8	8	4	3	11	17	8	7	8	8
Triangle-Directed	3	9	7	22	10	7	4	8	12	12	8	8	8	4	3	11	17	8	7	8	8
Tumor	13	18	9	13	12	13	15	16	9	9	16	20	20	16	13	23	3	12	13	11	20
Requited-Love	3	6	7	23	5	2	3	4	8	8	4	7	7	3	4	10	15	3	2	4	7
Vampire	4	4	8	24	6	3	3	3	8	8	3	6	6	3	5	9	16	5	3	4	6
Water-Flow	5	6	9	23	6	4	5	5	7	7	5	8	8	6	6	11	15	6	4	3	8
Weapon	10	6	15	31	12	10	7	7	16	16	7	3	3	6	10	4	24	11	10	11	3

Appendix B2-2

Mappings from the Assorted domains

Target domains are listed across the top of the following pages. The source domains run vertically down each of these pages.

Mapping	3-bears	Apple	Army	Arthurian-Saga	Assasinate-JFK	Assasinate-pig	Atom	Atom-Clone	Atom-Falkenhainer	Banker	Beautiful-Game	Bird	Burn-paper	Burn-rock	BUS	Buy-Apple	Canoeing	Caravaggio	Chair
3-Bears	3	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	0	1
Apple	3	4	4	4	2	2	1	1	1	1	3	3	1	1	2	1	1	3	3
Army	3	1	9	5	1	1	1	1	1	1	4	4	3	1	3	3	1	3	3
Arthurian-Saga	3	3	4	4	1	1	1	1	1	1	3	3	3	1	2	4	1	2	1
Assasinate-Jfk	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1
Assasinate-Pig	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Atom	1	1	2	2	2	2	2	2	3	1	2	2	2	1	1	1	2	1	1
Atom-Clone	1	1	2	2	2	2	2	2	3	1	2	2	2	1	1	1	2	1	1
Atom-Falkenhainer	1	1	3	3	3	3	2	2	4	1	3	3	2	1	1	1	3	1	1
Banker	3	1	1	2	1	1	1	1	1	2	2	2	1	1	2	2	1	2	2
Beautiful-Game	3	1	3	3	1	1	1	1	1	2	3	3	3	1	2	3	1	2	2
Bird	3	2	4	3	2	2	1	1	2	2	3	4	3	1	2	4	2	2	2
Burn-Paper	1	1	3	3	3	3	2	2	3	1	3	3	3	1	1	3	2	1	1
Burn-Rock	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bus	3	1	4	4	1	1	1	1	1	2	4	4	1	1	5	3	1	2	3
Buy-Apple	3	1	5	5	1	1	1	1	1	2	5	5	3	1	3	5	1	2	2
Canoeing	1	1	2	3	2	2	2	2	4	1	2	2	2	1	1	1	3	1	1
Caravaggio	3	2	2	6	1	1	1	1	1	2	2	2	3	1	3	2	1	13	4
Chair	3	3	3	1	1	1	1	1	1	2	2	2	1	1	3	2	1	2	7
Clothes	1	1	2	2	2	2	2	2	3	1	2	2	2	1	1	1	2	1	1
Composer	3	1	2	2	1	1	1	1	1	2	2	2	1	1	3	2	1	3	3
Computer	1	0	1	1	1	1	2	2	3	1	2	2	2	1	1	1	2	1	1
Create-Building	3	1	5	5	1	1	1	1	1	1	3	3	3	1	2	3	1	3	2
Cut-Apple	1	1	4	2	1	1	2	2	3	1	1	4	2	1	1	3	2	1	1
Cut-Ball	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cycling	1	1	3	3	3	3	2	2	3	1	3	3	3	1	1	3	2	1	1
Cycling2	1	1	4	2	1	1	2	2	3	1	1	3	2	1	1	3	2	1	1
Driving	3	1	3	3	3	3	2	2	3	1	3	3	3	1	2	3	2	1	1
Driving2	1	1	4	3	1	1	2	2	4	1	1	3	2	1	1	3	3	1	1
Eagle	3	2	3	3	1	1	1	1	1	2	2	2	1	1	3	2	1	2	5
Eat-Apple	1	1	4	2	2	2	2	2	3	1	2	4	2	1	1	3	2	1	1
Eat-Ball	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Fish	3	2	2	2	1	1	1	1	1	2	2	4	1	1	2	2	1	2	2
Flat-Ball	0	0	1	3	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0
Flower	3	2	3	3	2	2	2	2	3	2	3	3	2	1	3	2	2	2	3
Fortress	3	1	8	5	1	1	1	1	1	1	4	4	3	1	3	4	1	3	3
Fortress-Arcs	3	1	5	3	1	1	1	1	1	1	2	3	1	1	3	4	1	2	2
Fruit	3	1	2	2	1	1	1	1	1	1	2	2	1	1	2	2	1	2	1
Furniture	3	1	3	3	1	1	1	1	1	2	3	3	1	1	3	3	1	2	2
General	3	1	2	2	1	1	1	1	1	2	2	2	1	1	3	2	1	3	3
Golf-Play	3	1	3	3	1	1	1	1	1	2	3	3	3	1	2	3	1	1	2
Gun	3	3	3	2	1	1	1	1	1	2	2	2	1	1	3	2	1	3	5

Hammer	3	1	3	2	1	1	1	1	1	1	1	1	1	1	2	1	1	2	2
Heat-Flow	3	1	3	3	1	1	1	1	1	1	1	1	3	1	2	1	1	1	2
Heat-Flow-Good	3	1	3	3	1	1	1	1	1	2	2	2	3	1	3	2	1	3	3
Horse	3	3	4	3	1	1	1	1	1	2	2	3	1	1	3	2	1	4	3
House	3	2	1	2	1	1	1	1	1	2	2	2	1	1	2	2	1	2	2
Insect	3	1	2	2	1	1	1	1	1	1	2	3	1	1	2	1	1	2	1
John-Doe-Drive	1	1	3	1	1	1	1	1	1	1	1	3	1	1	1	3	1	1	1
Kennedy-Saga	3	3	4	4	1	1	1	1	1	1	3	3	3	1	2	4	1	2	1
Kick-About	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Knife-Cut	3	3	4	3	1	1	1	1	1	1	2	3	1	1	3	2	1	4	4
Lada-Car	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Leadbelly	3	1	4	5	1	1	1	1	1	2	2	2	3	1	3	2	1	12	5
Love-Triangle	3	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1
Man-Mind	3	1	2	2	1	1	1	1	1	2	2	2	1	1	2	2	1	2	2
Melt-Brick	3	1	2	2	1	1	1	1	1	2	2	2	1	1	2	2	1	2	2
Melt-Snow	3	2	4	4	1	1	1	1	1	2	4	4	3	1	2	4	1	2	2
Rectangle-Area	0	0	3	1	2	2	0	0	0	0	2	2	0	0	0	0	0	0	0
Rolls-Royce-Car	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Scissors-Cut	3	1	3	2	1	1	1	1	1	1	1	1	1	1	3	1	1	3	5
Seat-Drive	1	1	3	1	1	1	1	1	1	1	1	3	1	1	1	3	1	1	1
Shoe	3	2	3	3	1	1	1	1	1	1	2	2	1	1	3	2	1	2	2
Soccer	3	1	5	3	1	1	1	1	1	1	3	3	3	1	3	3	1	5	4
Solar-System	1	2	5	4	5	5	2	2	3	1	5	5	3	1	1	3	2	1	1
Solar-System-Falkenhainer	1	2	4	4	4	4	2	2	3	1	4	4	3	1	1	3	2	1	1
Spider	3	1	2	2	1	1	1	1	1	1	2	3	1	1	2	1	1	2	1
Story-Tell	1	1	3	3	1	1	2	2	3	1	3	3	3	1	1	3	2	1	1
Sun	3	2	7	5	5	5	2	2	3	1	5	7	3	1	2	2	1	1	1
Surgeon	3	1	3	3	1	1	1	1	1	2	2	2	1	1	3	2	1	3	3
Throw-Ball	1	1	3	3	3	3	2	2	3	1	3	3	3	1	1	3	2	1	1
Throw-Gun	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Throw-House	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Tool	3	1	2	2	2	2	2	2	3	1	2	2	2	1	2	1	2	1	1
Triangle	3	2	3	3	1	1	1	1	1	2	3	3	1	1	2	2	1	3	2
Triangle-Directed	3	2	3	3	1	1	1	1	1	2	3	3	1	1	2	2	1	3	2
Tumor	3	1	8	5	1	1	1	1	1	1	4	4	3	1	3	4	1	3	3
Requited-Love	3	1	3	3	1	1	1	1	1	2	3	3	3	1	2	3	1	1	2
Vampire	3	1	4	4	2	2	1	1	2	2	4	4	3	1	2	4	2	2	2
Water-Flow	3	2	4	5	1	1	1	1	1	2	3	3	3	1	2	2	1	2	2
Weapon	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Mapping	Clothes	Composer	Computer	Create-building	Cut-Apple	Cut-Ball	Cycling	Cycling2	Driving	Driving2	Eagle	Eat-Apple	Eat-Ball	Fish	Flat-Ball	Flower	Fortress	Fortress-ARCS	Fruit
3-Bears	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	1
Apple	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	3	1	1	1
Army	1	2	0	3	2	1	3	2	3	2	4	2	1	1	0	3	9	3	1
Arthurian-Saga	1	1	0	3	1	1	3	1	3	1	2	1	1	1	1	2	2	1	2
Assasinate-Jfk	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Assasinate-Pig	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Atom	2	1	1	1	2	1	2	2	2	2	1	2	1	1	0	2	1	1	1
Atom-Clone	2	1	1	1	2	1	2	2	2	2	1	2	1	1	0	2	1	1	1
Atom-Falkenhainer	2	1	1	1	2	1	2	2	2	3	1	2	1	1	0	2	1	1	1
Banker	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	1
Beautiful-Game	1	2	1	3	1	1	3	1	3	1	2	1	1	2	0	2	3	1	1
Bird	1	2	1	3	3	1	3	2	3	2	2	3	1	2	0	3	3	2	1
Burn-Paper	2	1	1	2	2	1	3	2	3	2	1	2	1	1	1	2	2	1	1
Burn-Rock	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Bus	1	3	1	1	1	1	1	1	1	1	3	1	1	2	0	3	2	2	1
Buy-Apple	1	2	1	3	2	1	3	2	3	2	3	2	1	2	0	2	5	3	2
Canoeing	2	1	1	1	2	1	2	2	2	3	1	2	1	1	0	2	1	1	1
Caravaggio	1	8	1	2	1	1	1	1	1	1	4	1	1	2	0	3	4	1	2
Chair	1	3	1	2	1	1	1	1	1	1	6	1	1	2	0	5	4	2	1
Clothes	2	1	1	1	2	1	2	2	2	2	1	2	1	1	0	2	1	1	1
Composer	1	8	1	2	1	1	1	1	1	1	3	1	1	2	0	2	2	2	1
Computer	2	1	1	1	2	1	2	2	2	2	1	2	1	1	0	2	0	1	1
Create-Building	1	2	0	5	1	1	3	1	3	1	3	1	1	1	0	2	4	2	1
Cut-Apple	2	1	1	1	3	1	2	3	2	3	1	3	1	1	0	2	2	2	1
Cut-Ball	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Cycling	2	1	1	2	2	1	3	2	3	2	1	2	1	1	0	2	2	1	1
Cycling2	2	1	1	1	3	1	2	3	2	3	1	3	1	1	0	2	3	2	1
Driving	2	1	1	3	2	1	3	2	4	1	3	2	1	1	0	4	3	1	1
Driving2	2	1	1	1	3	1	2	3	1	4	1	3	1	1	0	2	3	2	1
Eagle	1	2	1	2	1	1	1	1	2	1	6	1	1	2	0	5	2	1	1
Eat-Apple	2	1	1	1	3	1	2	3	2	3	1	3	1	1	0	2	2	2	1
Eat-Ball	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Fish	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	1
Flat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
Flower	2	1	1	1	2	1	2	2	2	2	4	2	1	2	0	5	3	1	1
Fortress	1	2	0	4	2	1	3	2	3	3	4	2	1	1	1	5	10	3	1
Fortress-Arcs	1	2	0	2	2	1	1	3	1	3	2	2	1	1	0	2	5	4	1
Fruit	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	2
Furniture	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	1
General	1	8	1	2	1	1	1	1	1	1	3	1	1	2	0	2	2	2	1
Golf-Play	1	2	1	3	1	1	3	1	3	1	2	1	1	2	1	2	2	1	1
Gun	1	5	1	2	1	1	1	1	1	1	4	1	1	2	0	3	4	2	1

Hammer	1	2	0	2	1	1	1	1	1	1	3	1	1	1	0	2	3	2	1
Heat-Flow	1	2	0	2	1	1	1	1	1	1	3	1	1	1	1	2	5	2	1
Heat-Flow-Good	1	3	1	2	1	1	1	1	1	1	3	1	1	2	1	3	2	2	1
Horse	1	3	1	2	1	1	1	1	1	1	3	1	1	2	0	3	2	2	1
House	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	1
Insect	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	2
John-Doe-Drive	1	1	0	1	2	1	1	2	1	2	1	2	1	1	0	1	2	1	1
Kennedy-Saga	1	1	0	3	1	1	3	1	3	1	2	1	1	1	1	2	2	1	2
Kick-About	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Knife-Cut	1	3	0	1	1	1	1	1	1	1	3	1	1	1	0	3	2	1	2
Lada-Car	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Leadbelly	1	8	1	2	1	1	1	1	1	1	4	1	1	2	0	2	5	2	1
Love-Triangle	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	1
Man-Mind	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	2	1	1
Melt-Brick	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	1
Melt-Snow	1	2	1	3	1	1	3	1	3	1	2	1	1	2	1	2	4	1	2
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Rolls-Royce-Car	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Scissors-Cut	1	2	0	2	1	1	1	1	1	1	6	1	1	1	0	4	2	2	1
Seat-Drive	1	1	0	1	2	1	1	2	1	2	1	2	1	1	0	1	2	1	1
Shoe	1	3	0	2	1	1	1	1	2	1	3	1	1	1	0	3	2	1	1
Soccer	1	4	0	3	1	1	3	1	3	1	3	1	1	1	1	3	5	2	1
Solar-System	2	1	1	3	2	1	3	2	3	2	1	2	1	1	0	2	2	1	1
Solar-System-Falkenhainer	2	1	1	2	2	1	3	2	3	2	1	2	1	1	1	2	2	1	1
Spider	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	2
Story-Tell	2	1	1	2	2	1	3	2	3	2	1	2	1	1	0	2	2	2	1
Sun	1	1	0	3	1	1	3	2	3	2	1	1	1	1	1	2	2	1	1
Surgeon	1	8	1	2	1	1	1	1	1	1	3	1	1	2	0	2	2	2	1
Throw-Ball	2	1	1	2	2	1	3	2	3	2	1	2	1	1	0	2	2	2	1
Throw-Gun	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Throw-House	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
Tool	2	1	1	1	2	1	2	2	2	2	1	2	1	1	0	4	1	1	1
Triangle	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	2
Triangle-Directed	1	2	1	1	1	1	1	1	1	1	2	1	1	2	0	2	1	1	2
Tumor	1	2	0	4	2	1	3	2	3	3	4	2	1	1	1	5	10	3	1
Requited-Love	1	2	1	3	1	1	3	1	3	1	2	1	1	2	1	2	2	1	1
Vampire	1	2	1	3	1	1	3	1	3	2	2	1	1	2	1	3	3	1	1
Water-Flow	1	2	1	1	1	1	1	1	1	1	2	1	1	2	1	2	5	1	2
Weapon	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

Mapping	Furniture	General	Golf-Play	Gun	Hammer	Heat-Flow	Heat-flow-Good	Horse	House	Insect	John-Doe-Drive	Kennedy-Saga	Kick-About	Knife-Cut	Lada-Car	Leadbelly	Love-triangle	Man-mind	Melt-brick	Melt-snow	Rectangle-area
3-Bears	1	1	1	2	1	1	0	1	1	2	1	0	1	2	1	1	3	1	1	1	0
Apple	1	1	1	3	1	1	0	2	2	3	1	3	1	3	1	3	3	1	1	2	0
Army	1	2	3	3	2	1	2	2	1	3	2	2	1	4	1	1	3	1	1	3	1
Arthurian-Saga	1	1	3	2	1	2	2	1	1	5	1	4	1	3	1	1	3	1	1	3	0
Assasinate-Jfk	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Assasinate-Pig	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Atom	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	2	1	1	1	1	0
Atom-Clone	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	2	1	1	1	1	0
Atom-Falkenhainer	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	2	1	1	1	1	0
Banker	2	2	2	2	1	1	0	2	2	2	1	0	1	2	1	2	3	2	2	2	0
Beautiful-Game	2	2	3	2	1	1	0	2	2	2	1	2	1	2	1	2	3	2	2	3	0
Bird	2	2	3	2	1	1	0	2	2	2	2	2	1	2	1	2	3	2	2	4	0
Burn-Paper	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	1	1	1	2	0
Burn-Rock	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Bus	3	3	2	3	1	1	0	2	2	3	1	0	1	4	1	2	3	2	2	2	0
Buy-Apple	3	2	3	2	1	1	0	2	2	2	2	3	1	2	1	2	3	2	2	4	0
Canoeing	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	2	1	1	1	1	0
Caravaggio	2	8	3	4	1	2	2	7	2	5	1	4	1	6	1	7	3	2	2	4	0
Chair	2	3	2	4	2	1	1	4	2	2	1	0	1	4	1	2	3	2	2	2	0
Clothes	2	1	1	1	1	1	0	1	1	1	1	0	1	1	1	2	1	1	1	1	0
Composer	2	8	2	3	2	1	1	6	2	3	1	0	1	3	1	2	3	2	2	2	0
Computer	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Create-Building	1	2	3	2	2	1	0	2	1	3	1	2	1	2	1	1	3	1	1	3	0
Cut-Apple	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	2	1	1	1	1	0
Cut-Ball	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Cycling	1	1	2	1	1	1	0	1	1	1	1	1	1	1	1	2	1	1	1	2	0
Cycling2	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	2	1	1	1	1	0
Driving	1	1	3	1	1	1	0	1	1	2	1	1	1	2	1	2	3	1	1	3	0
Driving2	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	2	1	1	1	1	0
Eagle	2	2	2	3	2	1	0	2	2	2	1	0	1	3	1	1	3	2	2	2	0
Eat-Apple	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	2	1	1	1	1	0
Eat-Ball	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Fish	2	2	2	2	1	1	0	2	2	2	1	0	1	2	1	1	3	2	2	2	0
Flat-Ball	0	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0
Flower	2	1	2	3	1	1	2	1	2	2	1	0	1	2	1	2	3	1	2	2	0
Fortress	1	2	3	4	3	2	2	2	1	3	2	2	1	4	1	1	3	1	1	3	1
Fortress-Arcs	1	2	1	3	2	1	2	2	1	3	2	0	1	3	1	2	3	1	1	1	1
Fruit	1	1	1	2	1	1	0	1	1	3	1	2	1	3	1	1	3	1	1	2	0
Furniture	3	2	2	2	1	1	0	2	2	2	1	0	1	2	1	2	3	2	2	2	0
General	2	8	2	3	2	1	1	6	2	3	1	0	1	3	1	2	3	2	2	2	0
Golf-Play	2	2	3	2	1	2	2	2	2	2	1	2	1	2	1	1	3	2	2	3	0
Gun	2	5	2	6	2	1	1	7	2	3	1	0	1	7	1	2	3	2	2	2	0

Hammer	1	2	1	2	3	1	0	2	1	2	1	0	1	3	1	2	3	1	1	1	0
Heat-Flow	1	2	3	2	2	5	4	2	1	3	1	2	1	2	1	3	3	1	1	3	0
Heat-Flow-Good	2	3	3	3	2	4	4	3	2	3	1	2	1	3	1	3	3	2	2	3	1
Horse	2	3	2	5	2	1	1	7	2	5	1	0	1	6	1	2	3	2	2	2	0
House	2	2	2	2	1	1	2	2	2	2	1	0	1	2	1	2	3	2	2	2	0
Insect	1	1	1	2	1	1	0	3	1	5	1	2	1	3	1	0	3	1	1	2	0
John-Doe-Drive	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	1	1	1	1	1	0
Kennedy-Saga	1	1	3	2	1	2	2	1	1	5	1	4	1	3	1	1	3	1	1	3	0
Kick-About	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Knife-Cut	2	3	1	5	3	1	1	7	2	5	1	2	1	9	1	1	3	1	1	2	0
Lada-Car	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Leadbelly	2	8	3	4	2	3	4	5	2	3	1	2	1	6	1	15	3	2	2	4	0
Love-Triangle	1	1	1	2	1	1	0	1	1	2	1	0	1	2	1	1	3	1	1	1	0
Man-Mind	2	2	2	2	1	1	0	2	2	2	1	0	1	2	1	1	3	2	2	2	0
Melt-Brick	2	2	2	2	1	1	0	2	2	2	1	0	1	2	1	2	3	2	2	2	0
Melt-Snow	2	2	3	2	1	2	2	2	2	3	1	1	1	2	1	2	3	2	2	4	0
Rectangle-Area	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Rolls-Royce-Car	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Scissors-Cut	1	2	1	3	2	1	1	2	1	2	1	0	1	2	1	1	3	1	1	1	0
Seat-Drive	1	1	1	1	1	1	0	1	1	1	2	0	1	1	1	1	1	1	1	1	0
Shoe	2	3	1	3	2	1	1	3	1	5	1	0	1	3	1	1	3	1	1	1	0
Soccer	1	4	3	3	2	2	2	3	1	3	1	2	1	3	1	3	3	1	1	3	0
Solar-System	1	1	2	1	1	1	0	1	1	1	1	1	1	1	1	2	1	1	1	3	0
Solar-System-Falkenhainer	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	2	1	1	1	2	0
Spider	1	1	1	2	1	1	0	3	1	3	1	2	1	3	1	0	3	1	1	2	0
Story-Tell	1	1	2	1	1	1	0	1	1	1	2	1	1	1	1	2	1	1	1	2	0
Sun	1	1	3	2	1	2	2	1	1	2	1	1	1	2	1	1	3	1	1	3	0
Surgeon	2	8	2	3	2	1	1	4	2	3	1	0	1	6	1	1	3	2	2	2	0
Throw-Ball	1	1	2	1	1	1	0	1	1	1	2	1	1	1	1	2	1	1	1	2	0
Throw-Gun	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Throw-House	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0
Tool	1	1	1	2	1	1	0	1	1	2	1	0	1	2	1	2	3	1	1	1	0
Triangle	2	2	2	2	1	1	2	2	2	3	1	2	1	3	1	2	3	2	2	3	1
Triangle-Directed	2	2	2	2	1	1	2	2	2	3	1	2	1	3	1	2	3	2	2	3	1
Tumor	1	2	3	4	3	3	3	2	1	3	2	2	1	4	1	2	3	1	1	3	0
Requited-Love	2	2	3	2	1	2	2	2	2	2	1	2	1	2	1	1	3	2	2	3	0
Vampire	2	2	3	2	1	2	2	2	2	2	1	2	1	2	1	2	3	2	2	4	0
Water-Flow	2	2	3	2	1	2	2	2	2	3	1	1	1	2	1	2	3	2	2	3	0
Weapon	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0

Mapping	Rolls-Royce-car	Scissors-cut	Seat-drive	Shoe	Soccer	Solar-system	Solar-system-Falkenhainer	Spier	Story-tell	Sun	Surgeon	Throw-ball	Throw-gun	Throw-house	Tool	Triangle	triangle-directed	Tumor	Requited-love	vampire	Water-flow	Weapon
3-Bears	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Apple	1	1	1	3	1	2	2	2	1	2	2	2	2	2	3	2	0	1	3	3	3	2
Army	1	3	2	4	3	3	3	2	3	4	4	4	1	1	1	1	0	9	4	4	2	1
Arthurian-Saga	1	1	1	1	3	3	3	3	3	3	3	3	1	1	2	2	0	2	3	3	4	1
Assasinate-Jfk	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Assasinate-Pig	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Atom	1	1	1	1	1	2	2	1	2	2	2	2	2	2	2	1	0	1	2	2	2	2
Atom-Clone	1	1	1	1	1	2	2	1	2	2	2	2	2	2	2	1	0	1	2	2	2	2
Atom-Falkenhainer	1	1	1	1	1	2	2	1	2	2	2	3	3	3	3	1	0	1	3	3	3	3
Banker	1	1	1	2	1	1	1	2	1	1	1	1	1	1	2	2	0	1	2	3	2	1
Beautiful-Game	1	1	1	2	2	3	3	2	3	3	3	3	1	1	3	2	0	3	3	3	3	1
Bird	1	1	2	1	2	3	3	2	3	2	2	3	2	2	4	2	0	3	3	4	4	2
Burn-Paper	1	1	1	1	2	3	3	1	3	3	3	3	3	3	3	1	0	2	3	3	3	3
Burn-Rock	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Bus	1	2	1	2	1	1	1	2	1	1	1	1	1	1	1	2	0	2	4	4	4	1
Buy-Apple	1	1	2	2	2	3	3	2	3	2	2	3	1	1	5	2	0	5	5	5	5	1
Canoeing	1	1	1	1	1	2	2	1	2	1	1	2	3	2	3	1	0	1	3	2	2	2
Caravaggio	1	3	1	3	4	1	3	3	1	2	2	1	1	1	2	3	0	4	4	4	5	1
Chair	1	3	1	5	1	1	1	1	1	1	1	1	1	1	1	2	0	4	2	2	2	1
Clothes	1	1	1	1	1	2	2	1	2	1	1	2	2	2	2	1	0	1	2	2	2	2
Composer	1	3	1	3	2	1	1	2	1	1	1	1	1	1	1	2	0	2	2	2	3	1
Computer	1	1	1	1	1	2	2	1	2	1	1	2	1	1	2	1	0	0	2	2	2	1
Create-Building	1	2	1	3	2	5	3	2	3	3	3	3	1	1	3	1	0	4	3	3	5	1
Cut-Apple	1	1	2	1	1	2	2	1	3	1	1	4	2	2	2	1	0	2	2	2	1	1
Cut-Ball	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Cycling	1	1	1	1	2	3	3	1	3	3	3	3	3	3	3	1	0	2	3	3	3	3
Cycling2	1	1	2	1	1	2	2	1	3	1	1	4	2	2	2	1	0	3	2	2	1	1
Driving	1	1	1	1	2	3	3	2	3	3	3	3	3	3	3	1	0	3	4	3	3	3
Driving2	1	1	2	1	1	2	2	1	3	1	1	4	3	2	3	1	0	3	3	2	1	1
Eagle	1	3	1	5	1	1	1	1	1	1	1	1	1	1	2	2	0	2	2	2	2	1
Eat-Apple	1	1	2	1	1	2	2	1	3	1	1	4	2	2	2	1	0	2	2	2	2	2
Eat-Ball	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Fish	1	1	1	1	1	1	1	2	1	1	1	1	1	1	2	2	0	1	2	2	2	1
Flat-Ball	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	0	0	1	3	3	3	1
Flower	1	2	1	2	1	2	2	1	2	1	1	2	2	2	2	2	0	3	3	3	3	2
Fortress	1	3	2	1	3	3	3	2	3	4	4	4	1	1	1	1	0	10	4	4	4	1
Fortress-Arcs	1	2	2	1	2	1	1	2	3	2	2	3	1	1	2	1	0	5	2	2	3	1
Fruit	1	1	1	1	1	1	1	3	1	1	1	1	1	1	2	2	0	1	2	2	2	1
Furniture	1	1	1	2	1	1	1	2	1	1	1	1	1	1	3	2	0	1	3	3	3	1
General	1	3	1	3	2	1	1	2	1	1	1	1	1	1	1	2	0	2	2	2	3	1

Golf-Play	1	1	1	2	2	3	3	2	3	3	3	3	1	1	3	2	0	2	3	3	3	1
Gun	1	3	1	3	1	1	1	2	1	1	1	1	1	1	1	2	0	4	2	2	3	1
Hammer	1	2	1	5	3	1	1	1	1	1	1	1	1	1	1	1	0	3	1	1	2	1
Heat-Flow	1	4	1	1	3	1	3	2	1	3	3	1	1	1	1	1	0	5	3	3	5	1
Heat-Flow-Good	1	2	1	2	3	1	3	2	1	3	3	1	1	1	1	2	0	5	3	3	5	1
Horse	1	2	1	5	1	1	1	3	1	1	1	1	1	1	2	2	0	2	2	2	3	1
House	1	1	1	2	1	1	1	2	1	1	1	1	1	1	2	2	0	1	2	2	2	1
Insect	1	1	1	3	1	1	1	3	1	1	1	1	1	1	2	2	0	1	2	2	2	1
John-Doe-Drive	1	1	2	1	1	1	1	1	3	0	0	3	1	1	1	1	0	2	1	1	1	1
Kennedy-Saga	1	1	1	1	3	3	3	3	3	3	3	3	1	1	2	2	0	2	3	3	4	1
Kick-About	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Knife-Cut	1	2	1	5	1	1	1	3	1	1	1	1	1	1	2	2	0	2	2	2	2	1
Lada-Car	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Leadbelly	1	3	1	5	3	1	3	2	1	2	2	2	2	2	2	2	0	5	4	5	5	1
Love-Triangle	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	0	1	2	1	1	1
Man-Mind	1	2	1	2	1	1	1	2	1	1	1	1	1	1	2	2	0	2	2	2	2	1
Melt-Brick	1	1	1	2	1	1	1	2	1	1	1	1	1	1	2	2	0	1	2	2	2	1
Melt-Snow	1	1	1	2	2	3	3	3	3	3	3	3	1	1	4	3	0	4	4	4	4	1
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	1	1	2	1	1	0	0	1	2	2	1
Rolls-Royce-Car	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
Scissors-Cut	1	4	1	5	2	1	1	1	1	1	1	1	1	1	1	1	0	2	1	1	2	1
Seat-Drive	1	1	2	1	1	1	1	1	3	0	0	3	1	1	1	1	0	2	1	1	1	1
Shoe	1	3	1	6	1	1	1	3	1	1	1	1	1	1	2	2	0	2	2	2	2	1
Soccer	1	3	1	1	10	3	3	2	3	3	3	3	1	1	1	1	0	5	3	3	4	1
Solar-System	1	1	1	1	2	5	3	1	3	5	5	5	5	5	5	1	0	2	4	4	5	5
Solar-System-Falkenhainer	1	1	1	1	2	3	4	1	3	3	3	4	4	4	4	1	0	2	4	4	4	4
Spider	1	1	1	1	1	1	1	3	1	1	1	1	1	1	2	2	0	1	2	2	2	1
Story-Tell	1	1	2	1	2	3	3	1	3	1	1	3	1	1	3	1	0	2	3	3	1	1
Sun	1	1	1	1	2	5	3	2	3	6	6	7	5	5	5	1	0	2	5	5	5	5
Surgeon	1	3	1	3	4	1	1	2	1	1	1	1	1	1	2	2	0	2	2	2	3	1
Throw-Ball	1	1	2	1	2	3	3	1	3	1	1	3	4	4	3	1	0	2	3	3	3	3
Throw-Gun	1	1	1	1	1	1	1	1	1	1	1	2	1	2	1	1	0	1	1	1	1	1
Throw-House	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	0	1	1	1	1	1
Tool	1	1	1	1	1	2	2	2	2	1	1	2	2	2	2	1	0	1	2	2	2	2
Triangle	1	1	1	2	1	1	1	3	1	1	1	1	1	1	3	3	0	1	3	3	3	1
Triangle-Directed	1	1	1	2	1	1	1	3	1	1	1	1	1	1	3	3	0	1	3	3	3	1
Tumor	1	3	2	1	3	3	3	2	3	4	4	4	1	1	1	1	0	10	4	4	4	1
Requited-Love	1	1	1	2	2	3	3	2	3	3	3	3	1	1	1	2	0	2	3	3	3	1
Vampire	1	1	1	2	2	3	3	2	3	3	3	3	2	2	4	2	0	3	4	4	4	2
Water-Flow	1	1	1	2	2	1	3	3	1	3	3	1	1	1	3	3	0	5	5	5	6	1
Weapon	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1

Appendix B2-3

Inferences from the Assorted domains

Total Infs	3-bears	Apple	Army	Arthurian-saga	Assasinate-JFK	Assasinate-pig	Atom	Atom-clone	atom-falkenhainer	Banker	Beautiful-game	Bird	Burn-paper	Burn-rock	Bus	Buy-apple	Canoeing	Caravaggio	Chair	Clothes
3-Bears	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Apple	2	0	3	1	1	1	1	1	1	2	2	1	1	1	3	2	1	1	1	1
Army	0	1	0	2	0	0	0	0	0	0	1	1	1	0	4	2	0	7	9	0
Arthurian-Saga	1	2	0	0	0	0	0	0	0	1	0	0	0	0	4	0	0	0	4	0
Assasinate-Jfk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Assasinate-Pig	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Atom	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	0
Atom-Clone	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	0
Atom-Falkenhainer	2	2	2	2	2	2	1	1	0	2	2	1	1	2	2	2	0	2	2	1
Banker	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Beautiful-Game	3	3	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	2	2	0
Bird	4	3	1	0	1	1	1	1	0	3	0	0	0	1	3	0	0	3	3	1
Burn-Paper	3	3	0	0	3	3	2	2	2	3	0	0	0	3	3	0	2	0	3	2
Burn-Rock	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus	0	3	2	3	0	0	0	0	0	2	2	2	0	0	0	1	0	0	2	0
Buy-Apple	5	5	1	1	0	0	0	0	0	4	2	1	0	0	3	0	0	4	4	0
Canoeing	1	1	1	2	1	1	0	0	0	1	1	0	0	1	1	1	0	2	1	0
Caravaggio	1	2	0	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
Chair	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Clothes	0	0	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	0
Composer	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Computer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create-Building	3	6	2	3	0	0	0	0	0	3	0	0	0	0	6	0	0	4	5	0
Cut-Apple	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
Cut-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cycling	3	3	0	0	3	3	2	2	2	3	0	0	0	3	3	0	2	0	3	2
Cycling2	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
Driving	2	3	0	0	3	3	2	2	2	3	0	0	0	3	3	0	2	3	3	2
Driving2	1	1	0	2	0	0	0	0	0	1	0	0	0	0	0	2	0	2	0	0
Eagle	1	1	1	2	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
Eat-Apple	1	2	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	0
Eat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fish	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flat-Ball	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flower	1	1	0	2	1	1	0	0	0	1	1	0	0	1	1	1	0	1	0	0
Fortress	0	2	0	2	0	0	0	0	0	0	1	1	0	0	4	1	0	4	9	0
Fortress-Arcs	2	2	0	2	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0
Fruit	0	2	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	1	0
Furniture	1	1	2	2	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
General	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Golf-Play	2	3	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	0	2	0
Gun	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Hammer	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Heat-Flow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Heat-Flow-Good	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Horse	1	0	2	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
House	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Insect	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0
John-Doe-Drive	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kennedy-Saga	1	2	0	0	0	0	0	0	0	1	0	0	0	0	4	0	0	0	4	0
Kick-About	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife-Cut	1	0	2	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	3	0
Lada-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Leadbelly	0	2	1	2	0	0	0	0	0	1	0	0	0	0	1	1	1	2	1	0
Love-Triangle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Man-Mind	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Melt-Brick	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Melt-Snow	3	3	0	1	0	0	0	0	0	3	1	0	0	0	3	0	0	0	3	0
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rolls-Royce-Car	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Scissors-Cut	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Seat-Drive	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoe	1	1	1	2	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0
Soccer	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
Solar-System	2	2	3	1	6	6	5	5	5	6	3	3	1	6	6	3	5	6	6	5
Solar-System-Falkenhainer	7	6	0	1	4	4	3	3	3	4	1	1	1	4	4	1	3	0	4	3
Spider	2	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0
Story-Tell	1	1	0	0	0	0	2	2	2	0	0	0	0	0	0	0	2	0	0	2
Sun	3	3	3	3	6	6	5	5	4	6	3	6	3	6	6	6	6	0	5	6
Surgeon	5	6	1	2	0	0	0	0	0	0	0	0	0	0	1	0	0	1	2	0
Throw-Ball	1	2	0	0	3	3	2	2	2	3	0	0	0	3	3	0	2	3	3	2
Throw-Gun	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Throw-House	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tool	1	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	1	1	0
Triangle	0	0	2	1	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0
Triangle-Directed	2	1	2	1	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	0
Tumor	2	2	0	2	0	0	0	0	0	0	1	1	0	0	4	1	0	4	9	0
Requited-Love	0	1	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	0	2	0
Vampire	1	2	1	1	1	1	1	1	0	3	1	0	0	1	3	0	0	0	3	1
Water-Flow	0	1	2	2	0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	0
Weapon	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Total Infs	Ccompser	Computer	Create-building	cut-apple	cut-ball	Cycling	Cycling2	Driving	Driving2	Eagle	Eat-apple	Eat-ball	Fish	Flat-ball	Flower	Fortress	Fortress-arcs	Fruit	Furniture	General
3-Bears	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Apple	3	0	3	1	1	1	1	2	1	3	1	1	2	0	2	3	3	2	2	3
Army	9	0	4	0	0	1	1	2	1	9	0	0	0	0	4	0	2	0	0	9
Arthurian-Saga	4	0	0	0	0	0	0	1	0	4	0	0	1	0	4	1	4	0	1	4
Assasinate-Jfk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Assasinate-Pig	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Atom	1	0	1	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1
Atom-Clone	1	0	1	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1
Atom-Falkenhainer	2	1	2	1	2	1	1	1	0	2	1	2	2	0	1	2	2	2	2	2
Banker	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Beautiful-Game	2	2	0	0	0	0	0	0	0	2	0	0	2	0	2	0	3	3	2	2
Bird	3	0	1	0	1	0	0	0	0	3	0	1	3	0	2	1	1	4	3	3
Burn-Paper	3	2	0	2	3	0	2	0	3	3	2	3	3	0	2	0	3	3	3	3
Burn-Rock	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus	2	0	3	0	0	0	0	0	0	2	0	0	2	0	2	2	2	0	1	2
Buy-Apple	4	0	2	0	0	0	0	2	0	4	0	0	4	0	4	0	1	4	3	4
Canoeing	1	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1
Caravaggio	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1
Chair	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3
Clothes	1	0	1	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	0	1
Composer	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Computer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create-Building	5	0	0	0	0	0	0	0	0	5	0	0	3	0	6	0	5	3	0	5
Cut-Apple	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1
Cut-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cycling	0	2	0	2	3	0	2	0	2	3	2	3	3	0	2	0	3	3	3	3
Cycling2	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Driving	2	2	0	2	3	0	2	0	0	3	2	3	3	0	2	0	3	3	3	3
Driving2	1	0	1	1	1	1	1	2	0	0	1	1	0	0	0	0	0	0	0	1
Eagle	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2	1	0	2
Eat-Apple	1	0	0	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1
Eat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fish	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0
Flat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flower	2	0	2	0	1	0	0	0	0	0	0	1	1	0	0	0	2	1	1	2
Fortress	9	0	5	0	0	1	1	1	0	9	0	0	0	0	3	0	5	0	0	9
Fortress-Arcs	1	0	1	0	0	0	0	1	0	2	0	0	1	0	2	0	0	1	1	1
Fruit	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1
Furniture	1	1	2	0	0	0	0	2	0	1	0	0	1	0	1	2	2	2	0	1
General	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Golf-Play	2	2	0	0	0	0	0	0	0	2	0	0	2	0	2	0	3	3	2	2
Gun	2	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	3
Hammer	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
Heat-Flow	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

Heat-Flow-Good	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
Horse	2	0	1	0	0	0	0	1	0	2	0	0	0	0	1	2	1	1	0	2
House	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Insect	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1
John-Doe-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kennedy-Saga	4	0	0	0	0	0	0	1	0	4	0	0	1	0	4	1	4	0	1	4
Kick-About	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife-Cut	3	0	2	0	0	0	0	1	0	2	0	0	1	0	0	2	2	0	0	5
Lada-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Leadbelly	1	0	1	1	1	1	1	0	1	2	1	1	1	0	1	1	0	0	0	3
Love-Triangle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Man-Mind	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0
Melt-Brick	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0
Melt-Snow	3	0	1	0	0	0	0	1	0	3	0	0	3	0	3	1	4	3	3	3
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rolls-Royce-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Scissors-Cut	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
Seat-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoe	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	2	1	0	1
Soccer	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	2	0	0	1
Solar-System	6	5	0	5	6	3	5	3	5	6	5	6	6	0	5	3	6	6	6	6
Solar-System-Falkenhainer	4	3	0	3	4	1	3	1	3	4	3	4	4	0	3	1	4	4	4	4
Spider	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1
Story-Tell	0	2	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
Sun	4	1	3	6	6	3	6	3	6	6	6	6	6	2	6	3	6	6	6	6
Surgeon	2	0	1	0	0	0	0	1	0	2	0	0	0	0	2	1	1	1	0	2
Throw-Ball	3	2	0	0	3	0	0	0	0	3	0	3	3	0	2	0	0	3	3	3
Throw-Gun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Throw-House	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tool	1	0	1	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1
Triangle	1	0	2	0	0	0	0	2	0	1	0	0	1	0	1	2	2	1	1	1
Triangle-Directed	1	0	2	0	0	0	0	2	0	1	0	0	1	0	1	2	2	1	1	1
Tumor	9	0	5	0	0	1	1	1	0	9	0	0	0	0	3	0	0	0	0	9
Requited-Love	2	2	0	0	0	0	0	0	0	2	0	0	2	0	2	0	0	0	2	2
Vampire	3	0	1	1	1	0	1	1	0	3	1	1	3	0	2	0	1	1	3	3
Water-Flow	1	0	2	0	0	0	0	2	0	1	0	0	1	0	1	1	2	1	1	1
Weapon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Total Infs	Golf-play	Gun	Hammer	Heat-flow	Heat-flow-good	Horse	House	Insect	John-doe-drive	Kennedy-saga	Kick-about	Knife-cut	Lada-car	Leadbelly	Love-triangle	Man-mind	Melt-brick	Melt-snow	Rectangle-area	Rolls-Royce-car
3-Bears	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Apple	2	1	3	3	3	2	0	3	1	1	1	1	1	0	2	1	2	1	0	1
Army	1	9	9	8	9	9	0	1	1	2	0	8	0	4	0	2	0	2	0	0
Arthurian-Saga	0	4	4	1	1	4	1	3	0	0	0	3	0	0	1	1	1	1	0	0
Assasinate-Jfk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Assasinate-Pig	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Atom	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1
Atom-Clone	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1
Atom-Falkenhainer	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	0	2
Banker	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Beautiful-Game	0	2	3	3	2	2	2	3	0	0	0	2	0	2	3	1	2	0	0	0
Bird	0	3	4	4	3	3	3	3	0	1	1	3	1	2	4	3	3	0	0	1
Burn-Paper	0	3	3	0	0	3	3	3	3	0	3	3	3	0	3	3	3	0	0	3
Burn-Rock	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus	2	2	3	3	2	2	2	3	0	3	0	2	0	1	0	2	2	2	0	0
Buy-Apple	2	4	5	5	4	4	4	5	0	1	0	4	0	4	5	1	4	1	0	0
Canoeing	1	1	1	1	1	1	1	1	2	2	1	1	1	1	2	0	1	1	0	1
Caravaggio	1	3	1	1	1	1	0	2	0	2	0	2	0	1	1	0	0	0	0	0
Chair	0	1	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0
Clothes	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1
Composer	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Computer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create-Building	0	5	5	5	4	5	3	6	0	0	0	5	0	2	3	3	3	0	0	0
Cut-Apple	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	0
Cut-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cycling	0	3	3	3	3	3	3	3	3	0	3	3	3	2	0	3	0	0	0	3
Cycling2	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1	0	0
Driving	0	3	3	3	3	3	4	3	3	0	3	3	3	2	4	3	3	0	0	3
Driving2	0	0	0	0	0	0	0	0	0	2	1	0	0	1	2	1	1	1	0	0
Eagle	0	1	1	2	1	1	0	2	0	1	0	1	0	1	1	0	0	0	0	0
Eat-Apple	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1
Eat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fish	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Flat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flower	1	1	2	2	1	1	1	2	1	2	1	1	1	1	1	1	1	1	0	1
Fortress	1	8	8	6	2	9	0	2	1	1	0	8	0	5	0	4	0	2	0	0
Fortress-Arcs	1	1	1	1	1	1	1	2	1	1	0	0	0	1	1	1	1	1	0	0
Fruit	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0
Furniture	1	1	2	2	1	1	1	2	0	1	0	1	0	1	2	1	1	1	0	0
General	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Golf-Play	0	2	3	0	0	2	2	3	0	0	0	2	0	0	3	2	2	0	0	0
Gun	0	0	2	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
Hammer	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Heat-Flow	0	2	0	0	0	2	0	0	0	0	0	2	0	2	0	0	0	0	0	0
Heat-Flow-Good	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Horse	0	0	2	2	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0

House	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Insect	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0
John-Doe-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kennedy-Saga	0	4	4	1	1	4	1	3	0	0	0	3	0	0	1	1	1	1	0	0
Kick-About	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife-Cut	1	3	2	3	1	1	0	1	0	1	0	0	0	2	1	1	1	0	0	0
Lada-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Leadbelly	0	1	1	1	0	1	0	1	0	2	1	0	0	0	1	0	1	1	0	0
Love-Triangle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Man-Mind	0	0	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0
Melt-Brick	0	0	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0
Melt-Snow	1	3	4	1	1	3	3	3	0	0	0	3	0	0	4	0	3	0	0	0
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rolls-Royce-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Scissors-Cut	0	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
Seat-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoe	1	1	1	2	1	1	1	1	0	1	0	1	0	0	1	0	1	1	0	0
Soccer	0	0	2	0	0	3	0	0	0	0	0	4	0	0	0	0	0	0	0	0
Solar-System	1	6	6	6	6	6	6	6	6	1	6	6	6	5	6	6	6	1	0	6
Solar-System-Falkenhainer	1	4	4	1	1	4	4	4	4	1	4	4	4	0	4	4	4	1	0	4
Spider	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0
Story-Tell	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Sun	3	6	6	3	3	6	6	6	7	3	6	6	6	3	6	6	6	3	0	6
Surgeon	0	1	1	1	0	1	0	2	0	2	0	1	0	1	1	0	0	0	0	0
Throw-Ball	0	3	3	3	3	3	3	3	0	0	3	3	3	2	3	3	3	0	0	3
Throw-Gun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Throw-House	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tool	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1
Triangle	1	1	2	2	1	1	1	1	0	1	0	1	0	1	2	1	1	0	0	0
Triangle-Directed	1	1	2	2	1	1	1	1	0	1	0	1	0	1	2	1	1	0	0	0
Tumor	1	8	8	6	6	9	0	2	1	1	0	8	0	5	0	4	0	2	0	0
Requited-Love	0	2	0	0	0	2	2	0	0	0	0	2	0	0	3	2	2	0	0	0
Vampire	1	3	1	1	1	3	3	1	1	1	1	3	1	0	4	3	3	0	0	1
Water-Flow	2	1	2	0	0	1	1	1	0	2	0	1	0	1	2	1	1	2	0	0
Weapon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Total Infs	Scissors-cut	Seat-drive	Shoe	Soccer	Solar-system	Solar-system-	Spier	Story-tell	Sun	Surgeon	Throw-ball	Throw-gun	Throw-house	Tool	Triangle	triangle-directed	Tumor	Requited-love	vampire	Water-flow	Weapon
3-Bears	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Apple	3	1	2	3	0	0	2	0	1	1	1	1	1	2	0	0	3	2	2	1	1
Army	9	1	9	5	0	1	0	1	1	3	1	0	0	0	1	0	0	1	1	0	0
Arthurian-Saga	4	0	4	1	0	0	0	0	1	2	0	0	0	1	0	0	1	0	0	1	0
Assasinate-Jfk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Assasinate-Pig	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Atom	1	1	1	1	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	1
Atom-Clone	1	1	1	1	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	1
Atom-Falkenhainer	2	2	2	2	1	1	2	1	1	1	1	2	2	1	1	0	2	2	1	2	2
Banker	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Beautiful-Game	3	0	2	0	0	0	2	0	0	1	0	0	0	3	2	0	0	0	0	2	0
Bird	4	0	4	0	0	0	2	0	1	2	0	1	1	4	3	0	0	0	0	3	1
Burn-Paper	3	3	3	0	0	0	2	0	0	1	0	3	3	2	0	0	0	0	0	0	3
Burn-Rock	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus	2	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0
Buy-Apple	5	0	4	2	0	0	4	0	2	3	0	0	0	5	1	0	0	2	1	4	0
Canoeing	1	1	1	1	0	0	1	0	1	1	0	2	1	1	0	0	0	2	0	1	1
Caravaggio	0	0	0	6	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0	1	0
Chair	1	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Clothes	1	1	1	1	0	0	1	0	1	1	0	1	1	0	0	0	1	1	0	1	1
Composer	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Computer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create-Building	5	0	5	3	0	0	2	0	0	2	0	0	0	3	3	0	0	0	0	5	0
Cut-Apple	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0
Cut-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cycling	3	3	3	0	0	0	2	0	0	1	0	3	3	2	0	0	0	0	0	3	3
Cycling2	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0
Driving	3	3	4	0	0	0	2	0	0	1	0	3	3	2	0	0	0	1	0	3	3
Driving2	0	0	0	0	0	0	0	0	0	0	1	2	1	1	0	0	0	2	0	0	0
Eagle	1	0	1	1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0
Eat-Apple	1	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	1
Eat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fish	1	0	1	1	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0
Flat-Ball	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flower	1	1	1	1	0	0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1
Fortress	9	1	10	6	1	0	0	1	1	3	1	0	0	0	0	0	0	1	1	0	0
Fortress-Arcs	1	1	2	1	0	0	1	0	1	1	0	0	0	1	1	0	0	1	1	1	0
Fruit	1	0	1	1	0	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	0
Furniture	2	0	1	2	0	0	2	0	2	2	0	0	0	2	1	0	2	1	1	1	0
General	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Golf-Play	3	0	2	0	0	0	2	0	0	1	0	0	0	3	2	0	0	0	0	0	0
Gun	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hammer	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Heat-Flow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
Heat-Flow-Good	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Horse	2	0	2	2	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0
House	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Insect	1	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	0
John-Doe-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kennedy-Saga	4	0	4	1	0	0	0	0	1	2	0	0	0	1	0	0	1	0	0	1	0
Kick-About	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Knife-Cut	0	0	2	5	0	0	0	0	1	1	0	0	0	1	0	0	2	1	1	0	0
Lada-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Leadbelly	2	0	0	8	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	0	0
Love-Triangle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Man-Mind	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0
Melt-Brick	1	0	0	1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0
Melt-Snow	4	0	3	1	1	0	2	0	1	2	0	0	0	4	2	0	0	1	0	1	0
Rectangle-Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rolls-Royce-Car	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Scissors-Cut	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
Seat-Drive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoe	1	0	0	2	0	0	0	0	1	1	0	0	0	1	0	0	1	1	1	1	0
Soccer	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Solar-System	6	6	6	1	0	0	4	3	2	3	3	6	6	5	0	0	0	1	1	6	6
Solar-System-Falkenhainer	4	4	4	1	0	0	3	0	0	1	1	4	4	3	0	0	0	1	1	1	4
Spider	1	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	0
Story-Tell	0	0	0	0	0	0	0	0	0	0	0	0	0	2	3	0	0	0	0	0	0
Sun	6	7	6	3	2	2	4	1	0	2	6	6	6	6	0	0	3	3	3	3	6
Surgeon	1	0	1	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0
Throw-Ball	3	0	3	0	0	0	2	0	0	1	0	3	3	2	0	0	0	0	0	3	3
Throw-Gun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Throw-House	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tool	1	1	1	1	0	0	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1
Triangle	2	0	1	2	0	0	1	0	2	2	0	0	0	2	0	0	2	1	1	0	0
Triangle-Directed	2	0	1	2	0	0	1	0	2	2	0	0	0	2	0	0	2	1	1	0	0
Tumor	9	1	10	6	1	0	0	1	1	3	1	0	0	0	0	0	0	1	1	0	0
Requited-Love	0	0	2	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0
Vampire	1	1	3	1	0	0	1	0	1	2	0	1	1	4	0	0	0	1	0	1	1
Water-Flow	2	0	1	2	0	0	1	0	2	2	0	0	0	2	0	0	1	2	2	0	0
Weapon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix C

WordNet Definitions

This appendix lists definitions for relations, objects and attributes used throughout the thesis. These definitions have been obtained from WordNet 1.7.1 online, unless otherwise stated.

Drive

Results for “Hypernyms (this is one way to...)” search of verb “drive”

21 senses of drive

Sense 1
drive -- (operate or control a vehicle; “drive a car or bus”; “Can you drive this four-wheel truck?”)
=> operate, control -- (handle and cause to function; “do not operate machinery after imbibing alcohol”; “control the lever”)
=> manipulate -- (hold something in one's hands and move it)
=> handle, palm -- (touch, lift, or hold with the hands)
=> touch -- (make physical contact with, come in contact with; “Touch the stone for good luck”; “She never touched her husband”)

Sense 2
drive, motor -- (travel or be transported in a vehicle; “We drove to the university every morning”; “They motored to London for the theater”)
=> travel, go, move, locomote -- (change location; move, travel, or proceed; “How fast does your new car go?”; “We travelled from Rome to Naples by bus”; “The policemen went from door to door looking for the suspect”;”The soldiers moved towards the city in an attempt to take it before night fell”)

Sense 3
drive -- (cause someone or something to move by driving; “She drove me to school every day”; “We drove the car to the garage”)
=> move, displace -- (cause to move, both in a concrete and in an abstract sense; “Move those boxes into the corner, please”; “I'm moving my money to another bank”; “The director moved more responsibilities onto his new assistant”)
Also See-> drive out#2; drive out#1; drive off#1; drive away#1

Sense 4

create -- (invest with a new title, office, or rank; "Create one a peer")
      => appoint, charge --
 (assign a duty, responsibility or obligation to; "He was appointed deputy manager"; "
She was charged with supervising the creation of a concordance")

The remaining entries have not been included in this Appendix.
Create:

6 senses of create

Sense 1
make, create --
 (make or cause to be or to become; "make a mess in one's office"; "create a furor")

Sense 2
create --
 (bring into existence; "The company was created 25 years ago"; "He created a new m
ovement in painting")

Sense 3
create --
 (pursue a creative activity; be engaged in a creative activity; "Don't disturb him--
he is creating")
      => act, move --
 (perform an action, or work out or perform (an action); "think before you act"; "We
must move quickly"; "The governor should act on the new energy bill"; "The nanny a
cted quickly by grabbing the toddler and covering him with a wet towel")

Sense 4
create -- (invest with a new title, office, or rank; "Create one a peer")
      => appoint, charge --
 (assign a duty, responsibility or obligation to; "He was appointed deputy manager"; "
She was charged with supervising the creation of a concordance")

The remaining entries have not been included in this Appendix.

Touch:

6 senses of create

Sense 1
make, create --
 (make or cause to be or to become; "make a mess in one's office"; "create a furor")

Sense 2
create --
 (bring into existence; "The company was created 25 years ago"; "He created a new m
ovement in painting")

Sense 3
create --
(pursue a creative activity; be engaged in a creative activity; "Don't disturb him--he is creating")
=> act, move --
(perform an action, or work out or perform (an action); "think before you act"; "We must move quickly"; "The governor should act on the new energy bill"; "The nanny acted quickly by grabbing the toddler and covering him with a wet towel")

Sense 4
create -- (invest with a new title, office, or rank; "Create one a peer")
=> appoint, charge --
(assign a duty, responsibility or obligation to; "He was appointed deputy manager"; "She was charged with supervising the creation of a concordance")

The remaining entries have not been included in this Appendix.

Appendix D1

Inferences classified as Valid by Kilaza using the Assorted KB
All duplicate entries have been removed from this data.

Attack Army Fortress
Control Man Car
Cut Human Something
Found Apple Tree
Holds John Apple
Holds John Apple
Jealous-Of Tom Joe
Opposite-Sign Nucleus Electron
Own Man Car
Throw Tom Football
Affect Architect Builder
Affect Caravaggio Paintbrush
Affect Human Object
Affect Joe-Kennedy Democratic-Nomination
Affect Merlin Excalibur
Affect Tom Joe
Attack Architect House
Attack Army Society
Attack Human Blade
Attack Human Hammer-Handle
Attack Human Vehicle
Attack Musician Composer
Attack Orchestra Listenership
Attack Patient Surgeon
Attack Sick-People Operating-Procedure
Attack Soldier General
Attracts Animal Fruit
Attracts Apple Apple-Core
Attracts Apple John
Attracts Apple-Core Apple
Attracts Architect Blue-Print
Attracts Banker Money
Attracts Barrell Gun
Attracts Bike Man
Attracts Blade Knife
Attracts Bob Football
Attracts Bob Rain
Attracts Brain Man
Attracts Brick House
Attracts Bus Wheel
Attracts Camp-Fire Rock
Attracts Cannoe Man
Attracts Car John-Doe
Attracts Car Lada
Attracts Car Lada
Attracts Car Man
Attracts Car Rolls-Royce
Attracts Car Seat
Attracts Caravaggio Paintbrush
Attracts Chair Wood
Attracts Composer Listenership
Attracts Daddy-Bear Mommy-Bear
Attracts Eagle Eagle-Head
Attracts Eagle-Head Eagle
Attracts Fish Water
Attracts Foot Shoe
Attracts Football Bob
Attracts Football Field
Attracts Football Tom
Attracts Footer Ground
Attracts Fortress Roads
Attracts Fortress Swamp
Attracts Fruit Animal
Attracts Furniture Human
Attracts General Society
Attracts Golf-Ball Golf-Green
Attracts Gun Barrell
Attracts Gun Tom
Attracts Hammer-Handle Hammer-Head
Attracts Hammer-Head Hammer-Handle
Attracts Healthy-Tissue Tumour
Attracts Horse Human
Attracts House Brick
Attracts House Tom
Attracts Human Clothes
Attracts Human Furniture
Attracts Human Horse
Attracts Insect Insect-Head
Attracts Insect-Head Insect
Attracts Iron-Bar Coffee
Attracts Jfk White-House
Attracts Joe-Kennedy Jfk
Attracts John Shop
Attracts John Sun
Attracts John-Doe Car
Attracts Knife Blade
Attracts Lada Car
Attracts Listenership Composer
Attracts Man Brain
Attracts Mary Sun
Attracts Mary Tom
Attracts Merlin Arthur
Attracts Mommy-Bear Daddy-Bear
Attracts Money Banker
Attracts Operating-Procedure Surgeon
Attracts Orange Orange-Core
Attracts Orange-Core Orange
Attracts Person Weapon
Attracts Pig Pig-House
Attracts Pig-House Pig
Attracts Program Cpu
Attracts Rain Bob
Attracts Roads Fortress
Attracts Rock Camp-Fire
Attracts Rolls-Royce Car
Attracts Scissors Something
Attracts Seat Car
Attracts Shoe Foot
Attracts Shop John
Attracts Sky Bird
Attracts Society General
Attracts Something Scissors
Attracts Stem Flower
Attracts Story Bob
Attracts Sun John
Attracts Surgeon Operating-Procedure
Attracts Swamp Fortress
Attracts Temperature-A Temperature-B
Attracts Tom Football
Attracts Tom Gun
Attracts Tom House
Attracts Tom Mary
Attracts Tool Human
Attracts Vehicle Wheel
Attracts Water Beaker
Attracts Water Fish
Attracts Weapon Person
Attracts Wheel Bus
Attracts Wheel Vehicle
Attracts White-House Jfk
Attracts Wood Chair
Avoid Army Fortress
Avoid Army General
Avoid Army Society
Avoid Builder House
Avoid Human Blade
Avoid Human Hammer-Head
Avoid Joe-Kennedy Democratic-Nomination
Avoid John Shop
Avoid Mary Sun
Avoid Merlin Excalibur
Avoid Music Leadbelly
Avoid Orchestra Composer
Avoid Orchestra Listenership
Avoid Player Team
Avoid Sick-People Operating-Procedure
Avoid Sick-People Surgeon
Become Barrell Metal
Become Blade Wood
Become Eagle-Head Eagle-Torso
Become Foot Shoe-Sole
Become Fortress Obj_Fortress
Become Hammer-Handle Hammer
Become Human Horse-Head
Become Insect-Head Insect-Legs
Become Listenership Orchestra
Become Operating-Procedure Patient
Become Society Army
Become Something Scisssors
Become Stem Veg-Substance
Become Wheel Human
Become Wood Chair-Back
Born-In Keeper Laces
Born-In Surgeon Crash-Cart
Born-In Surgeon Junior-Surgeon
Born-In Team Goal
Build Human Shoe-Sole
Build Player Team
Burn Animal Fruit
Burn Apple-Core Apple
Burn Banker Money
Burn Bob Football
Burn Bob Rain
Burn Brain Man
Burn Bus Wheel
Burn Camp-Fire Rock
Burn Chair Wood
Burn Composer Listenership
Burn Daddy-Bear Mommy-Bear
Burn Eagle Eagle-Head
Burn Fish Water
Burn Furniture Human
Burn General Society
Burn Gun Barrell
Burn Hammer-Head Hammer-Handle
Burn Horse Human
Burn House Brick
Burn Insect Insect-Head
Burn Jfk White-House
Burn John Sun

Burn John-Doe Car
Burn Knife Table
Burn Lada Car
Burn Lada Car
Burn Orange Tree
Burn Pig Pig-House
Burn Roads Fortress
Burn Rolls-Royce Car
Burn Scissors Something
Burn Seat Car
Burn Shoe Foot
Burn Surgeon Operating-Procedure
Burn Tom Football
Burn Tom Gun
Burn Tom House
Burn Tom Mary
Burn Vehicle Wheel
Burn Weapon Person
Buy Human Object
Buy Tom Joe
Connect Animal Fruit
Connect Apple Tree
Connect Apple-Core Tree
Connect Apple-Core Veg-Substance
Connect Architect Blue-Print
Connect Architect House
Connect Army Path
Connect Army Soldier
Connect Army Sword
Connect Artillery Soldier
Connect Banker Money
Connect Barrell Handle
Connect Blood Coffin
Connect Blue-Print Builder
Connect Bob Football
Connect Bob Rain
Connect Brick Human
Connect Bus Wheel
Connect Camp-Fire Rock
Connect Car Man
Connect Coffee Iron-Bar
Connect Coffee Temperature-A
Connect Daddy-Bear Mommy-Bear
Connect Field Goal-Keeper
Connect Field Player
Connect Fin Water
Connect Fish Water
Connect Football Team
Connect Footer Ground
Connect Fortress Army
Connect Fortress Path
Connect Fortress Road
Connect Furniture Human
Connect Golf-Ball Golf-Green
Connect Golf-Green Flag
Connect Ground Wall
Connect Gun Barrell
Connect Handle Barrell
Connect Healthy-Tissue Path
Connect Heat Coffee
Connect House Human
Connect Human Barrell
Connect Human House
Connect Human Vehicle-Body
Connect Ice-Cube Temperature-B
Connect Jfk White-House
Connect Joe-Kennedy Jfk
Connect Joe-Kennedy President
Connect John Shop
Connect John Sun
Connect John-Doe Car
Connect King Arthur
Connect Lada Car
Connect Listenership Conductor-Baton
Connect Mary Joe
Connect Mary Sun
Connect Medical-Asistants Patient
Connect Merlin Arthur
Connect Merlin King
Made-Of Merlin King
Connect Mommy-Bear Baby-Bear
Connect Money Bank
Connect Obj_Fortress Fortress
Connect Operating-Procedure Sick-People
Connect Orange Tree
Connect Orange-Core Tree
Connect Orange-Core Veg-Substance
Connect Orange-Peel Apple
Connect Orange-Peel Orange-Core
Connect Orchestra Conductor-Baton
Connect Orchestra Musician
Connect Paintbrush Bristle
Connect Path Beam
Connect Path Platoon
Connect Pig Pig-House
Connect Planet Sun
Connect Player Goal-Keeper
Connect President Jfk
Connect Road Platoon
Located-In Platoon Swamp
Revolves Fortress Platoon
Connect Roads Fortress
Connect Roads Obj_Fortress
Connect Rolls-Royce Car
Connect Scalpel Medical-Asistants
Connect Seat Car
Connect Seat Wheel
Connect Shop Apple
Connect Shop Apple
Connect Sick-People Medical-Asistants
Connect Sick-People Patient
Connect Society Sword
Connect Sun Brick
Connect Sun Habitation
Connect Surgeon Patient
Connect Swamp Path
Connect Swamp Road
Connect Sword Artillery
Connect Team Goal-Keeper
Connect Temperature-A Ice-Cube
Connect Temperature-A Temperature-B
Connect Temperature-B Coffee
Connect Temperature-B Coffee
Connect Temperature-B Coffee
Connect Temperature-B Coffee
Connect Tom Football
Connect Tom Gun
Connect Tom House
Connect Tom Mary
Connect Tom Mary
Connect Tool Object
Connect Tumour Path
Connect Vehicle Wheel
Connect Vehicle-Body Wheel
Connect Water Beaker
Connect Weapon Person
Connect Wheel Seat
Connect Wheel Vehicle-Body
Connect Wings Sky
Conquer Apple Orange-Core
Conquer Army General
Conquer Army Roads
Conquer Blade Scissors
Conquer Chair Chair-Leg
Conquer Coffee Temperature-A
Conquer Eagle Wings
Conquer Football Team
Conquer Gun Bullet
Conquer Hammer Hammer-Head
Conquer Horse-Head Horse
Conquer Human Shoe
Conquer Line P
Conquer Orchestra Composer
Conquer Paintbrush Caravaggio
Conquer Sick-People Surgeon
Conquer Temperature-A Heat
Conquer Tone Leadbelly
Contains Apple-Core Tree
Contains Architect House
Contains Army General
Contains Hammer-Head Human
Contains Horse Human
Contains Insect Insect-Legs
Contains Joe-Kennedy President
Contains Merlin King
Contains Orange-Core Tree
Contains Orchestra Composer
Contains Roads Army
Contains Sick-People Surgeon
Contains Temperature-A Ice-Cube
Control Architect Blue-Print
Control Architect Builder
Control Banker Money
Control Bob Football
Control Bob Rain
Control Joe-Kennedy Jfk
Control Joe-Kennedy President
Control John Apple
Control John Sun
Control Keeper Team
Control Leadbelly Human
Control Mary Sun
Control Merlin Arthur
Control Merlin King
Control Tom Football
Control Tom Gun
Control Tom House
Control Tom Mary
Converge Barrell Something
Converge Blue-Print House
Converge Chair-Seat Chair-Leg
Converge Eagle-Head Wings
Converge Hammer Hammer-Handle
Converge Horse-Torso Human
Converge Human Wheel
Converge Insect-Legs Insect-Head
Converge Musician Listenership
Converge Note Guitar
Converge Orange-Peel Orange
Converge Patient Operating-Procedure
Converge Roads Fortress
Converge Scisssors Something
Converge Shoe-Sole Foot
Converge Soldier Society
Converge Something Blade
Converge Tree Apple
Create General Sword
Create Human Metal
Create Human Seat
Create Joe-Kennedy Democratic-Nomination
Create Keeper Clubs
Create Merlin Excalibur
Create Team Pitch
Cut Architect Blue-Print
Cut Artillery Cannon
Cut Banker Money
Cut Banker Money
Cut Bob Football
Cut Bob Rain
Cut Caravaggio Paintbrush
Cut General Society
Cut Human Wall
Cut Joe-Kennedy Jfk

Cut John Sun
Cut Mary Sun
Cut Merlin Arthur
Cut Surgeon Operating-Procedure
Cut Tom Football
Cut Tom Gun
Cut Tom House
Cut Tom Mary
Damage Chair Chair-Legs
Damage Civilian Artillery
Damage Coffee Iron-Bar
Damage Composer Drum
Damage Flesh Human
Damage Flesh Wings
Damage Foot Shoe-Upper
Damage Football Boots
Damage General Cannon
Damage Human Chair-Legs
Damage Human Wood
Damage Leadbelly Human
Damage Musician Conductor-Baton
Damage Orange-Peel Veg-Substance
Damage Patient Medical-Asistants
Damage Platoon Road
Connect Swamp Road
Damage Soldier Sword
Damage Something Scisssors
Damage Surgeon Medical-Staff
Decorate Animal Fruit
Decorate Animal Fruit-Seed
Decorate Apple-Core Apple
Decorate Apple-Core Veg-Substance
Decorate Architect Blue-Print
Decorate Architect House
Decorate Banker Money
Decorate Bob Football
Decorate Bob Rain
Decorate Brain Man
Decorate Bus Wheel
Decorate Camp-Fire Rock
Decorate Caravaggio Paintbrush
Decorate Chair Wood
Decorate Composer Listenership
Decorate Daddy-Bear Baby-Bear
Decorate Daddy-Bear Mommy-Bear
Decorate Eagle Eagle-Head
Decorate Engine Man
Decorate Fish Water
Decorate Football Field
Decorate Football Player
Decorate Footer Ground
Decorate Fortress Road
Decorate Fortress Swamp
Decorate General Society
Decorate Golf-Ball Golf-Green
Decorate Gun Barrell
Decorate Hammer-Head Hammer-Handle
Decorate Hammer-Head Human
Decorate Healthy-Tissue Path
Decorate Healthy-Tissue Tumour
Decorate Horse Human
Decorate House Brick
Decorate Human Object
Decorate Insect Insect-Body
Decorate Insect Insect-Head
Decorate Iron-Bar Coffee
Decorate Jfk White-House
Decorate Joe-Kennedy Jfk
Decorate John Shop
Decorate John Sun
Decorate John-Doe Car
Decorate Knife Blade
Decorate Lada Car
Decorate Mary Sun
Decorate Merlin Arthur
Decorate Merlin Excalibur
Decorate Orange-Core Orange
Decorate Orange-Core Veg-Substance
Decorate Pig Pig-House
Decorate Planet Habitation
Decorate Planet Sun
Decorate Roads Army
Decorate Roads Fortress
Decorate Rolls-Royce Car
Decorate Scissors Blade
Decorate Scissors Something
Decorate Seat Car
Decorate Shoe Foot
Decorate Surgeon Operating-Procedure
Decorate Swamp Fortress
Decorate Swamp Path
Decorate Temperature-A Coffee
Decorate Temperature-A Temperature-B
Decorate Tom Football
Decorate Tom Gun
Decorate Tom House
Decorate Tom Joe
Decorate Tom Mary
Decorate Vehicle Wheel
Decorate Water Beaker
Decorate Weapon Person
Died General Weapon
Died Keeper Goal
Died Surgeon Instruments
Died Surgeon Junior-Surgeon
Died Team Keeper
Directed-Line Animal Fruit-Seed
Directed-Line Apple-Core Veg-Substance
Directed-Line Architect Builder
Directed-Line Barrell Metal
Directed-Line Blood Coffin
Directed-Line Blue-Print Builder
Directed-Line Brick Human
Directed-Line Car Man
Directed-Line Coffee Iron-Bar
Directed-Line Daddy-Bear Baby-Bear
Directed-Line Eagle-Head Flesh
Directed-Line Engine Man
Directed-Line Field Player
Directed-Line Foot Shoe-Sole
Directed-Line Football Player
Directed-Line Fortress Army
Directed-Line Fortress Path
Directed-Line Fortress Road
Line Swamp Road
Directed-Line Golf-Green Flag
Directed-Line Ground Wall
Directed-Line Guitar Black-Musicans
Directed-Line Hammer-Handle Human
Directed-Line Hammer-Head Human
Directed-Line Healthy-Tissue Path
Directed-Line Human Flesh
Directed-Line Human House
Directed-Line Human Object
Directed-Line Insect Insect-Body
Directed-Line Joe-Kennedy Democratic-Nomination
Directed-Line Knife Knife-Handle
Directed-Line Listenership Orchestra
Directed-Line Man Mind
Directed-Line Mary Joe
Directed-Line Mary Joe
Directed-Line Merlin Excalibur
Directed-Line Mommy-Bear Baby-Bear
Directed-Line Money Bank
Directed-Line Operating-Procedure
Directed-Line Orange-Core Veg-Substance
Directed-Line Planet Habitation
Directed-Line Roads Army
Directed-Line Scissors Blade
Directed-Line Shop Apple
Directed-Line Sky Wings
Directed-Line Society Army
Directed-Line Something Blade
Directed-Line Stem Veg-Substance
Directed-Line Sun Brick
Directed-Line Sun Habitation
Directed-Line Swamp Path
Directed-Line Temperature-A Coffee
Directed-Line Temperature-B Coffee
Directed-Line Tom Joe
Directed-Line Tool Object
Directed-Line Tumour Path
Directed-Line Water Fin
Directed-Line Wheel Human
Directed-Line Wood Chair-Seat
Drive Caravaggio Paintbrush
Drive Human Brick
Drive Human Foot
Drive Joe Mary
Drive Joe-Kennedy Jfk
Drive John Apple
Drive Leadbelly Guitar
Drive Man Bike
Drive Merlin Arthur
Drive Surgeon Scalpel
Drive Tom Gun
Drive Tom Mary
Examine Apple Orange
Examine Army Fortress
Examine Army Society
Examine Baby-Bear Mommy-Bear
Examine Bank Money
Examine Blade Something
Examine Brick Sun
Examine Bullet Barrell
Examine Chair-Back Wood
Examine Fin Water
Examine Flower-Bloom Stem
Examine Fruit-Seed Fruit
Examine Horse-Head Human
Examine Human Brick
Examine Human Hammer-Handle
Examine Ice-Cube Temperature-B
Examine Insect-Body Insect-Head
Examine Joe Mary
Examine Mind Man
Examine Object Tool
Examine Orchestra Listenership
Examine P P
Examine Seat Wheel
Examine Sick-People Operating-Procedure
Examine Temperature-B Coffee
Examine Veg-Substance Apple
Examine Vehicle-Body Wheel
Examine Vial Beaker
Expose Animal Fruit-Seed
Expose Apple-Core Veg-Substance
Expose Architect Builder
Expose Daddy-Bear Baby-Bear
Expose Engine Man
Expose Football Player
Expose Fortress Road
Expose Hammer-Head Human
Expose Healthy-Tissue Path
Expose Human Object
Expose Insect Insect-Body
Expose Joe-Kennedy Democratic-Nomination
Expose Merlin Excalibur
Expose Orange-Core Veg-Substance

Expose Planet Habitation
Expose Roads Army
Expose Scissors Blade
Expose Swamp Path
Expose Temperature-A Coffee
Expose Tom Joe
Flies-Through Animal Fruit
Flies-Through Tom House
Flow-Along Apple Tree
Flow-Along Chair-Legs Chair-Back
Flow-Along Civilian Army
Flow-Along Horse-Legs Horse-Head
Flow-Along Human Metal
Flow-Along Listener Orchestra
Flow-Along Medical-Asistants Sick-People
Flow-Along Note Violence
Flow-Along Table Wood
Flow-From Blood Vampire
Flow-From Democratic-Nomination Joe-Kennedy
Flow-From Excalibur Merlin
Flow-From Flag Golf-Ball
Flow-From Habitation Sun
Flow-From Joe Tom
Flow-From Line P
Flow-From Player Football
Flow-From Sun Mary
Flow-To Animal Fruit-Seed
Flow-To Apple Veg-Substance
Flow-To Apple-Core Veg-Substance
Flow-To Architect Builder
Flow-To Army Fortress
Flow-To Army Path
Flow-To Blood Coffin
Flow-To Chair-Legs Chair-Leg
Flow-To Civilian Sword
Flow-To Daddy-Bear Baby-Bear
Flow-To Democratic-Nomination Jfk
Flow-To Engine Man
Flow-To Excalibur Arthur
Flow-To Flag Golf-Green
Flow-To Fortress Road
Flow-To Habitation Planet
Flow-To Hammer-Head Human
Flow-To Horse-Legs Flesh
Flow-To Human Handle
Flow-To Human Object
Flow-To Insect Insect-Body
Flow-To Joe Mary
Flow-To Listener Conductor-Baton
Flow-To Note Black-Musicans
Flow-To Orange-Core Veg-Substance
Flow-To Player Field
Flow-To Roads Army
Flow-To Scissors Blade
Flow-To Sun Snow
Flow-To Table Knife-Handle
Flow-To Tom Joe
Flow-To X-Ray Path
Flow-To X-Ray Tumour
Go_Down Apple Orange-Core
Go_Down Army General
Go_Down Baby-Bear Daddy-Bear
Go_Down Bank Banker
Go_Down Blade Scissors
Go_Down Brick John
Go_Down Builder Architect
Go_Down Chair-Back Chair
Go_Down Coffin Vampire
Go_Down Democratic-Nomination Joe-Kennedy
Go_Down Excalibur Merlin
Go_Down Fin Fish
Go_Down Flag Golf-Ball
Go_Down Flesh Eagle
Go_Down Flower-Bloom Flower
Go_Down Fruit-Seed Animal
Go_Down Habitation Planet
Go_Down Horse-Head Horse
Go_Down House Furniture
Go_Down Human Hammer-Head
Go_Down Human House
Go_Down Human Shoe
Go_Down Ice-Cube Temperature-A
Go_Down Insect-Body Insect
Go_Down Joe Tom
Go_Down Man Engine
Go_Down Mind Brain
Go_Down Note Leadbelly
Go_Down Object Human
Go_Down Orchestra Composer
Go_Down P P
Go_Down Painting Caravaggio
Go_Down Player Football
Go_Down Sick-People Surgeon
Go_Down Snow Mary
Go_Down Temperature-A Heat
Go_Down Veg-Substance Apple-Core
Go_Down Vehicle-Body Vehicle
Go_Down Vial Water
Go_Down Wall Footer
Go-To Animal Fruit-Seed
Go-To Architect Builder
Go-To Engine Man
Go-To Football Player
Go-To Footer Wall
Go-To Golf-Ball Flag
Go-To Joe-Kennedy Democratic-Nomination
Go-To Tom Joe
Greater Architect Builder
Greater Eagle Eagle-Torso
Greater Hammer-Head Human
Greater Human Shoe
Greater Insect-Body Insect-Legs
Greater Orange-Core Veg-Substance
Greater Seat Bus
Greater Tree Apple-Core
Greater Vehicle-Body Vehicle
Greater-Pressure Barrell Metal
Greater-Pressure Blade Wood
Greater-Pressure Blue-Print Builder
Greater-Pressure Brick Human
Greater-Pressure Car Man
Greater-Pressure Eagle-Head Flesh
Greater-Pressure Foot Shoe-Sole
Greater-Pressure Fortress Army
Greater-Pressure Ground Wall
Greater-Pressure Guitar Black-Musicans
Greater-Pressure Hammer-Handle Human
Greater-Pressure Human Flesh
Greater-Pressure Human House
Greater-Pressure Listenership Orchestra
Greater-Pressure Man Mind
Greater-Pressure Mary Joe
Greater-Pressure Mommy-Bear Baby-Bear
Greater-Pressure Money Bank
Greater-Pressure Operating-Procedure
Greater-Pressure Paintbrush Italian-School
Greater-Pressure Shop Apple
Greater-Pressure Sky Wings
Greater-Pressure Society Army
Greater-Pressure Something Blade
Greater-Pressure Stem Veg-Substance
Greater-Pressure Sun Brick
Greater-Pressure Swamp Road
Greater-Pressure Tool Object
Greater-Pressure Water Fin
Greater-Pressure Wheel Human
Greater-Pressure Wood Chair-Seat
Has Barrell Handle
Has Bristle Non-Derivitive
Has Chair-Seat Chair-Leg
Has Democratic-Nomination President
Has Eagle-Head Eagle-Torso
Has Excalibur King
Has Goal-Keeper Football
Has Horse-Legs Horse-Torso
Has Insect Insect-Body
Has Musician Conductor-Baton
Has Patient Operating-Procedure
Has Scisssors Blade
Has Soldier Sword
Has Stem Flower-Bloom
Has Veg-Substance Tree
Has Veg-Substance Tree
Has Wheel Seat
Has-Part Animal Fruit-Seed
Has-Part Apple John
Has-Part Architect Builder
Has-Part Army Fortress
Has-Part Arthur Merlin
Has-Part Baby-Bear Mommy-Bear
Has-Part Bank Money
Has-Part Beaker Water
Has-Part Bike Man
Has-Part Blood Vampire
Has-Part Blue-Print Architect
Has-Part Brick Sun
Has-Part Bus Seat
Has-Part Cannoe Man
Has-Part Car Engine
Has-Part Car John-Doe
Has-Part Car Lada
Has-Part Car Lada
Has-Part Car Man
Has-Part Car Rolls-Royce
Has-Part Car Seat
Has-Part Chair Chair-Back
Has-Part Coffee Temperature-B
Has-Part Coffin Blood
Has-Part Composer Conductor-Baton
Has-Part Conductor-Baton Listenership
Has-Part Daddy-Bear Baby-Bear
Has-Part Eagle Eagle-Torso
Has-Part Electron Nucleus
Has-Part Engine Man
Has-Part Excalibur Arthur
Has-Part Field Football
Has-Part Flag Golf-Green
Has-Part Flower Flower-Bloom
Has-Part Flower-Bloom Flower
Has-Part Flower-Bloom Veg-Substance
Has-Part Football Bob
Has-Part Football Player
Has-Part Football Team
Has-Part Football Tom
Has-Part Fortress Army
Has-Part Fortress Road
Has-Part Fortress Roads
Has-Part Fortress Swamp
Has-Part Fruit-Seed Fruit
Has-Part General Sword
Has-Part Goal-Hanger Clubs
Has-Part Golf-Green Golf-Ball
Has-Part Ground Footer
Has-Part Gun Human
Has-Part Gun Tom
Has-Part Habitation Sun
Has-Part Hammer Hammer-Head

Has-Part Hammer-Handle Hammer-Head
Has-Part Hammer-Head Human
Has-Part Healthy-Tissue Path
Has-Part House Tom
Has-Part Human Barrell
Has-Part Human Brick
Has-Part Human Clothes
Has-Part Human Furniture
Has-Part Human Hammer-Head
Has-Part Human Object
Has-Part Iron-Bar Coffee
Has-Part Jfk Joe-Kennedy
Has-Part Joe Mary
Has-Part Joe-Kennedy President
Has-Part Listenership Composer
Has-Part Man Brain
Has-Part Mary Tom
Has-Part Merlin Excalibur
Has-Part Merlin King
Has-Part Mommy-Bear Daddy-Bear
Has-Part Money Banker
Has-Part Object Tool
Has-Part Operating-Procedure Surgeon
Has-Part Paintbrush Caravaggio
Has-Part Paper Candle
Has-Part Person Weapon
Has-Part Pig-House Pig
Has-Part Planet Habitation
Has-Part Player Field
Has-Part Rain Bob
Has-Part Roads Army
Has-Part Scissors Blade
Has-Part Seat Wheel
Has-Part Shop John
Has-Part Sick-People Operating-Procedure
Has-Part Snow Sun
Has-Part Society General
Has-Part Something Blade
Has-Part Something Scissors
Has-Part Sun John
Has-Part Sun Mary
Has-Part Swamp Army
Has-Part Swamp Fortress
Has-Part Swamp Path
Has-Part Sword Society
Has-Part Temperature-A Coffee
Has-Part Temperature-A Iron-Bar
Has-Part Temperature-B Temperature-A
Has-Part Tom Joe
Has-Part Tool Human
Has-Part Tumour Healthy-Tissue
Has-Part Vial Beaker
Has-Part Wall Ground
Has-Part Wheel Bus
Has-Part White-House Jfk
Has-Part Wood Chair-Back
Hoards Architect Builder
Hoards Army Road
Hoards Joe-Kennedy President
Hoards Merlin King
Hoards Tom Joe
Holds Army Obj_Fortress
Holds Army Road
Holds Army Swamp
Holds Human Something
Holds Tom Football
Influence Architect House
Influence Human Handle
Influence Human Object
Influence Human Seat
Influence Joe-Kennedy President
Influence Merlin King
Influence Team Field
Influence Tom Joe
Inhabits Army Swamp
Keep-Out Junior-Surgeon Scalpel
Lead_To Car John-Doe
Lead_To Car Seat
Lifestyle Keeper Studs
Lifestyle Team Pitch
Line Animal Fruit-Seed
Line Apple-Core Veg-Substance
Line Architect Builder
Line Barrell Metal
Line Blood Coffin
Line Blue-Print Builder
Line Brick Human
Line Car Man
Line Coffee Iron-Bar
Line Daddy-Bear Baby-Bear
Line Eagle-Head Flesh
Line Engine Man
Line Field Player
Line Foot Shoe-Sole
Line Football Player
Line Fortress Army
Line Fortress Path
Line Fortress Road
Has Platoon Army
Line Golf-Green Flag
Line Ground Wall
Line Guitar Black-Musicans
Line Hammer-Handle Human
Line Hammer-Head Human
Line Healthy-Tissue Path
Line Human Flesh
Line Human House
Line Human Object
Line Insect Insect-Body
Line Joe-Kennedy Democratic-Nomination
Line Knife Knife-Handle
Line Listenership Orchestra
Line Man Mind
Line Mary Joe
Line Merlin Excalibur
Line Mommy-Bear Baby-Bear
Line Money Bank
Line Operating-Procedure Sick-People
Line Orange-Core Veg-Substance
Line Planet Habitation
Line Roads Army
Line Scissors Blade
Line Shop Apple
Line Sky Wings
Line Society Army
Line Something Blade
Line Stem Veg-Substance
Line Sun Brick
Line Sun Habitation
Line Swamp Path
Line Temperature-A Coffee
Line Tom Joe
Line Tool Object
Line Tumour Path
Line Water Fin
Line Wheel Human
Line Wood Chair-Seat
Live-In Architect Builder
Live-In Human Object
Live-In Joe-Kennedy Democratic-Nomination
Live-In Merlin Excalibur
Live-In Tom Joe
Located-In Apple Orange
Located-In Army Society
Located-In Baby-Bear Mommy-Bear
Located-In Bank Money
Located-In Black-Musicans Guitar
Located-In Blade Something
Located-In Blue-Print Builder
Located-In Brick Sun
Located-In Car Man
Located-In Chair-Seat Wood
Located-In Coffee Temperature-B
Located-In Coffin Blood
Located-In Field Player
Located-In Fin Water
Located-In Flesh Human
Located-In Golf-Green Flag
Located-In Ground Wall
Located-In Habitation Sun
Located-In House Human
Located-In Human Brick
Located-In Human Hammer-Handle
Located-In Human Wheel
Located-In Insect-Body Insect-Head
Located-In Iron-Bar Coffee
Located-In Italian-School Paintbrush
Located-In Joe Mary
Located-In Mary Joe
Located-In Metal Barrell
Located-In Mind Man
Located-In Obj_Fortress Roads
Located-In Object Tool
Located-In Orchestra Listenership
Located-In P P
Located-In Sick-People Operating-Procedure
Located-In Snow Sun
Located-In Veg-Substance Apple
Located-In Veg-Substance Stem
Located-In Vial Beaker
Located-In Wings Sky
Located-In Wood Blade
Lusts-After Banker Money
Lusts-After Bob Football
Lusts-After Bob Rain
Lusts-After General Society
Lusts-After Human Tool
Lusts-After John Apple
Lusts-After John Sun
Lusts-After Man Bike
Lusts-After Surgeon Operating-Procedure
Lusts-After Tom Football
Lusts-After Tom Gun
Lusts-After Tom House
Lusts-After Tom Mary
Made-Of Animal Fruit-Seed
Made-Of Architect House
Made-Of Army General
Made-Of Army Society
Made-Of Army Sword
Made-Of Blade Blade
Made-Of Blade Something
Made-Of Blood Coffin
Made-Of Blue-Print Builder
Made-Of Car Man
Made-Of Coffee Iron-Bar
Made-Of Daddy-Bear Baby-Bear
Made-Of Daddy-Bear Baby-Bear
Made-Of Engine Man
Made-Of Field Player
Made-Of Fish Water
Made-Of Foot Human
Made-Of Football Player
Made-Of Football Team
Made-Of Fortress Army
Made-Of Fortress Path
Made-Of Fortress Road
Made-Of Fruit-Seed Animal
Made-Of Golf-Green Flag

Made-Of Ground Wall
Made-Of Hammer-Handle Human
Made-Of Hammer-Head Hammer
Made-Of Healthy-Tissue Beam
Made-Of Healthy-Tissue Path
Made-Of Human House
Made-Of Human Object
Made-Of Human Shoe
Made-Of Human Shoe-Upper
Made-Of Human Wall
Made-Of Insect Insect-Body
Made-Of Insect Insect-Legs
Made-Of Insect-Head Insect-Legs
Made-Of Joe-Kennedy President
Made-Of Listenership Conductor-Baton
Made-Of Mary Joe
Made-Of Merlin Excalibur
Made-Of Merlin King
Made-Of Mommy-Bear Baby-Bear
Made-Of Money Bank
Made-Of Operating-Procedure Sick-People
Made-Of Orchestra Composer
Made-Of Orchestra Conductor-Baton
Made-Of Orchestra Listenership
Made-Of Planet Habitation
Made-Of Roads Army
Made-Of Roads Obj_Fortress
Made-Of Scissors Scisssors
Made-Of Shop Apple
Made-Of Sick-People Medical-Asistants
Made-Of Sick-People Operating-Procedure
Made-Of Sick-People Surgeon
Made-Of Society Sword
Made-Of Something Blade
Made-Of Sun Brick
Made-Of Sun Habitation
Made-Of Swamp Path
Made-Of Swamp Platoon
Made-Of Swamp Road
Made-Of Temperature-A Coffee
Made-Of Temperature-A Ice-Cube
Made-Of Temperature-A Iron-Bar
Made-Of Temperature-B Coffee
Made-Of Tom Joe
Made-Of Tool Object
Made-Of Tumour Path
Made-Of Vehicle Vehicle-Body
Made-Of Wheel Seat
Make Architect Blue-Print
Make Banker Money
Make Bob Football
Make Bob Rain
Make Caravaggio Paintbrush
Make General Society
Make Joe-Kennedy Jfk
Make John Shop
Make John Sun
Make Mary Sun
Make Merlin Arthur
Make Surgeon Operating-Procedure
Make Tom Football
Make Tom Gun
Make Tom House
Make Tom Mary
Next-To Composer Listener
Next-To Eagle Wings
Next-To Football Goal-Keeper
Next-To General Civilian
Next-To Human Shoe-Upper
Next-To Surgeon Patient
Next-To Temperature-A Iron-Bar
Obtain Foot Human
Obtain Human Flesh
Obtain Human House
Obtain Man Mind
Obtain Mary Joe
Obtain Mommy-Bear Baby-Bear
On-Top-Of Roads Fortress
Opposite-Sign Animal Fruit
Opposite-Sign Apple-Core Apple
Opposite-Sign Architect Blue-Print
Opposite-Sign Banker Money
Opposite-Sign Bird Sky
Opposite-Sign Bob Football
Opposite-Sign Bob Rain
Opposite-Sign Bob Story
Opposite-Sign Brain Man
Opposite-Sign Bus Wheel
Opposite-Sign Camp-Fire Rock
Has-Part Rock Camp-Fire
Opposite-Sign Caravaggio Paintbrush
Opposite-Sign Chair Wood
Opposite-Sign Clothes Human
Opposite-Sign Composer Listenership
Opposite-Sign Cpu Program
Opposite-Sign Daddy-Bear Mommy-Bear
Opposite-Sign Eagle Eagle-Head
Opposite-Sign Engine Car
Opposite-Sign Fish Water
Opposite-Sign Flower Stem
Opposite-Sign Football Field
Opposite-Sign Footer Ground
Opposite-Sign Fortress Swamp
Directed-Line Swamp Road
Opposite-Sign Furniture Human
Opposite-Sign General Society
Opposite-Sign Golf-Ball Golf-Green
Opposite-Sign Gun Barrell
Opposite-Sign Hammer-Head Hammer-Handle
Opposite-Sign Healthy-Tissue Tumour
Opposite-Sign Horse Human
Opposite-Sign House Brick
Opposite-Sign Human Tool
Opposite-Sign Insect Insect-Head
Opposite-Sign Iron-Bar Coffee
Opposite-Sign Jfk White-House
Opposite-Sign Joe-Kennedy Jfk
Opposite-Sign John Apple
Opposite-Sign John Shop
Opposite-Sign John Sun
Opposite-Sign John-Doe Car
Opposite-Sign Knife Blade
Opposite-Sign Lada Car
Opposite-Sign Leadbelly Guitar
Opposite-Sign Man Bike
Opposite-Sign Mary Sun
Opposite-Sign Merlin Arthur
Opposite-Sign Orange-Core Orange
Opposite-Sign P P
Opposite-Sign Pig Pig-House
Opposite-Sign Roads Fortress
Opposite-Sign Rolls-Royce Car
Opposite-Sign Scissors Something
Opposite-Sign Seat Car
Opposite-Sign Shoe Foot
Opposite-Sign Sun Earth
Opposite-Sign Sun Planet
Opposite-Sign Surgeon Operating-Procedure
Opposite-Sign Swamp Fortress
Opposite-Sign Temperature-A Temperature-B
Opposite-Sign Tom Football
Opposite-Sign Tom Gun
Opposite-Sign Tom House
Opposite-Sign Tom Mary
Opposite-Sign Vampire Blood
Opposite-Sign Vehicle Wheel
Opposite-Sign Water Beaker
Opposite-Sign Weapon Person
Own Caravaggio Paintbrush
Own General Army
Own Surgeon Scalpel
Owns Banker Money
Owns Bob Football
Owns Bob Rain
Owns Bob Story
Owns Caravaggio Paintbrush
Owns General Society
Owns Human Tool
Owns Joe-Kennedy Jfk
Owns John Apple
Owns John Shop
Owns John Sun
Owns Man Bike
Owns Man Cannoe
Owns Man Car
Owns Mary Sun
Owns Merlin Arthur
Owns Surgeon Sick-People
Owns Team Studs
Owns Tom Football
Owns Tom Gun
Owns Tom House
Owns Tom Mary
Owns Vampire Blood
Paddle Caravaggio Paintbrush
Paddle Human Tool
Paddle Joe-Kennedy Jfk
Paddle John Apple
Paddle Leadbelly Guitar
Paddle Man Bike
Paddle Merlin Arthur
Paddle Tom Gun
Paddle Tom Mary
Paddle Tom Mary
Part-Of Animal Fruit-Seed
Part-Of Apple Veg-Substance
Part-Of Apple-Core Veg-Substance
Part-Of Architect Blue-Print
Part-Of Architect Builder
Part-Of Architect House
Part-Of Bird Wings
Part-Of Blade Knife-Handle
Part-Of Blade Wood
Part-Of Bob Story
Part-Of Builder Architect
Part-Of Candle Paper
Part-Of Daddy-Bear Baby-Bear
Part-Of Democratic-Nomination President
Part-Of Engine Car
Part-Of Engine Man
Part-Of Excalibur King
Part-Of Flesh Horse-Legs
Part-Of Footer Ground
Part-Of Golf-Ball Golf-Green
Part-Of Hammer-Head Hammer-Handle
Part-Of Hammer-Head Human
Part-Of Hammer-Head Wall
Part-Of Heat Temperature-A
Part-Of Horse Horse-Torso
Part-Of Horse-Head Flesh
Part-Of Horse-Head Horse-Torso
Part-Of Human Hammer-Handle
Part-Of Human Object
Part-Of Human Shoe-Sole
Part-Of Human Shoe-Upper
Part-Of Human Vehicle-Body
Part-Of Insect-Body Insect-Legs
Part-Of Insect-Head Insect-Body

Part-Of Insect-Head Insect-Legs
Part-Of Iron-Bar Temperature-A
Part-Of Joe-Kennedy Democratic-Nomination
Part-Of Joe-Kennedy Jfk
Part-Of John Apple
Part-Of John-Doe Car
Part-Of Knife Knife-Handle
Part-Of Knife-Handle Wood
Part-Of Man Bike
Part-Of Man Car
Part-Of Mary Snow
Part-Of Merlin Arthur
Part-Of Merlin Excalibur
Part-Of Orange-Core Veg-Substance
Part-Of P Line
Part-Of Planet Habitation
Part-Of Roads Army
Part-Of Roads Obj_Fortress
Part-Of Seat Car
Part-Of Shoe Shoe-Sole
Part-Of Shoe Shoe-Upper
Part-Of Shoe-Sole Human
Part-Of Shoe-Sole Shoe-Upper
Part-Of Shoe-Upper Shoe-Sole
Part-Of Sun Earth
Part-Of Sun Mary
Part-Of Temperature-A Coffee
Part-Of Tom Football
Part-Of Tom Joe
Part-Of Tom Mary
Part-Of Vampire Coffin
Part-Of Veg-Substance Tree
Part-Of Vehicle-Body Human
Part-Of Wood Knife-Handle
Revolves Apple John
Revolves Arthur Merlin
Revolves Barrell Gun
Revolves Beaker Water
Revolves Bike Man
Revolves Blade Knife
Revolves Blue-Print Architect
Revolves Brick House
Revolves Cannoe Man
Revolves Car Engine
Revolves Car John-Doe
Revolves Car Lada
Revolves Car Man
Revolves Car Rolls-Royce
Revolves Car Seat
Revolves Coffee Iron-Bar
Revolves Coffin Vampire
Revolves Eagle-Head Eagle
Revolves Electron Nucleus
Revolves Field Football
Revolves Foot Shoe
Revolves Football Bob
Revolves Football Tom
Revolves Fortress Platoon
Revolves Fortress Roads
Revolves Fruit Animal
Revolves Fruit Animal
Revolves Golf-Green Golf-Ball
Revolves Ground Footer
Revolves Guitar Leadbelly
Revolves Gun Tom
Revolves Hammer-Handle Hammer-Head
Revolves House Tom
Revolves Human Clothes
Revolves Human Furniture
Revolves Human Horse
Revolves Insect-Head Insect
Revolves Jfk Joe-Kennedy
Revolves John-Doe Car
Revolves Listenership Composer
Revolves Man Brain
Revolves Man Brain
Revolves Mary Tom
Revolves Mommy-Bear Daddy-Bear
Revolves Money Banker
Revolves Operating-Procedure Surgeon
Revolves Orange Orange-Core
Revolves Paintbrush Caravaggio
Revolves Paper Candle
Opposite-Sign Candle Paper
Revolves Person Weapon
Revolves Pig-House Pig
Revolves Platoon Army
Revolves Rain Bob
Revolves Rock Camp-Fire
Revolves Seat Car
Revolves Snow Sun
Revolves Society General
Revolves Something Scissors
Revolves Stem Flower
Revolves Sun John
Revolves Temperature-B Temperature-A
Revolves Tool Human
Revolves Tumour Beam
Revolves Vial Beaker
Revolves Water Fish
Revolves Wheel Bus
Revolves Wheel Vehicle
Revolves White-House Jfk
Revolves Wings Bird
Revolves Wood Chair
See Architect Builder
See Merlin Excalibur
Go-To Merlin Excalibur
See Platoon Army
See Tom Joe
Shoots-With Human Hammer-Head
Shoots-With Human Leadbelly
Shoots-With Human Shoe
Shoots-With Player Team
Sit-In Human Hammer-Head
Sit-In Human House
Sit-In Human Vehicle
Sit-In Joe Tom
Sit-In King Merlin
Sit-In Man Brain
Sit-In President Joe-Kennedy
Sit-On Army Civilian
Sit-On Orchestra Listener
Sit-On Sick-People Medical-Assistants
Sit-On Team Goal
Split-Into Apple Orange-Peel
Split-Into Army Soldier
Split-Into Black-Musicans Note
Split-Into Blade Scisssors
Split-Into Brain Mind
Split-Into Builder Architect
Split-Into Bus Wheel
Split-Into Chair Chair-Seat
Split-Into Coffee Ice-Cube
Split-Into Eagle Eagle-Head
Split-Into Flower Stem
Split-Into Gun Barrell
Split-Into Hammer Wall
Split-Into Heat Temperature-A
Split-Into Horse-Head Horse-Torso
Split-Into Human Hammer
Split-Into Human Shoe-Upper
Split-Into Human Something
Split-Into Iron-Bar Temperature-A
Split-Into Music Note
Split-Into Orchestra Musician
Split-Into Paintbrush Bristle
Split-Into Player Football
Split-Into Shoe-Upper Shoe-Sole
Split-Into Sick-People Patient
Split-Into Sun Mary
Split-Into Temperature-A Temperature-B
Split-Into Tone Note
Subject-Of Human Chair-Back
Subject-Of Knife-Handle Human
Subject-Of Metal Something
Subject-Of Operating-Procedure Medical-Staff
Subject-Of Veg-Substance Orange-Peel
Surround Human Chair-Leg
Surround Joe Mary
Surround Team Field
Tell P P
Throw Animal Fruit
Throw Apple-Core Apple
Throw Banker Money
Throw Bob Football
Throw Bob Rain
Throw Brain Man
Throw Bus Wheel
Throw Camp-Fire Rock
Throw Caravaggio Paintbrush
Throw Chair Wood
Throw Composer Listenership
Throw Daddy-Bear Mommy-Bear
Throw Eagle Eagle-Head
Throw Fish Water
Throw Furniture Human
Throw General Society
Throw Gun Barrell
Throw Hammer-Head Hammer-Handle
Throw Horse Human
Throw House Brick
Throw Insect Insect-Head
Throw Iron-Bar Coffee
Throw Jfk White-House
Throw John Sun
Throw Knife Blade
Throw Lada Car
Throw Lada Car
Throw Orange-Core Orange
Throw Pig Pig-House
Throw Rolls-Royce Car
Throw Scissors Something
Throw Shoe Foot
Throw Surgeon Operating-Procedure
Throw Temperature-A Temperature-B
Throw Tom Gun
Throw Tom House
Throw Tom Mary
Throw Vehicle Wheel
Throw Water Beaker
Throw Weapon Person
Thud Animal Fruit-Seed
Thud Apple-Core Veg-Substance
Thud Daddy-Bear Baby-Bear
Thud Hammer-Head Wall
Thud Human Object
Thud Insect Insect-Body
Thud Orange-Core Veg-Substance
Thud Roads Army
Thud Scissors Blade
Thud Temperature-A Coffee
Thud Tom Joe
Transport Architect House
Transport Army General
Transport Army Swamp
Transport Army Swamp
Flow-To Fortress Road
Transport Banker Bank
Transport Human Shoe

Transport Joe-Kennedy President
Transport John Brick
Transport Mary Snow
Transport Merlin King
Transport Orchestra Composer
Transport Sick-People Surgeon
Transport Team Field
Transport Tom Joe
Transport Vampire Coffin
Type-Of Cannon Weapon
Type-Of Drum Musical-Instrument
Use Artillery General
Use Human Bullet
Use Human Gun
Use Human Seat
Use Human Vehicle-Body
Use Keeper Football
Used-For Coffee Temperature-B
Used-For Guitar String
Used-For Human Horse-Head
Used-For Scalpel Operating-Procedure
Used-In Animal Fruit-Seed
Used-In Apple-Core Veg-Substance
Used-In Architect House
Used-In Banker Bank
Used-In Bird Wings
Used-In Brain Mind
Used-In Caravaggio Italian-School
Used-In Chair Chair-Seat
Used-In Clothes Human
Used-In Composer Orchestra
Used-In Cpu Program
Used-In Daddy-Bear Baby-Bear
Used-In Eagle Flesh
Used-In Engine Man
Used-In Fish Fin
Used-In Flower Veg-Substance
Used-In Football Player
Used-In Footer Wall
Used-In Fortress Road
Used-In General Army
Used-In Golf-Ball Flag
Used-In Gun Metal
Used-In Hammer-Head Human
Used-In Healthy-Tissue Path
Used-In Horse Flesh
Used-In House Human
Used-In Human Object
Used-In Insect Insect-Body
Used-In Joe-Kennedy Democratic-Nomination
Used-In John Brick
Used-In Knife Wood
Used-In Leadbelly Black-Musicans
Used-In Mary Snow
Used-In Merlin Excalibur
Used-In Orange-Core Veg-Substance
Used-In P P
Used-In Planet Habitation
Used-In Roads Army
Used-In Scissors Blade
Used-In Shoe Shoe-Sole
Used-In Surgeon Sick-People
Used-In Swamp Path
Used-In Temperature-A Coffee
Used-In Temperature-A Iron-Bar
Used-In Tom Joe
Used-In Tom Joe
Used-In Vampire Coffin
Used-In Water Vial
Works-In Architect Builder
Works-In Human Object
Works-In Joe-Kennedy Democratic-Nomination
Works-In Merlin Excalibur
Works-In Tom Joe

Appendix D2

This lists the inferences rejected by Kilaza from the Assorted KB. The data lists the rejected predicate, and the predicate role (ie agent or patient) that was violated.

Rejected Predicate Role Violated
Affect Healthy-Tissue X-Ray #-Agnt
Attack Note Leadbelly #-Agnt
Attracts P P #-Ptnt
Avoid Black-Musicans Leadbelly #-Agnt
Avoid Paintbrush Italian-School #-Agnt
Avoid Tone Guitar #-Agnt
Born-In P Line #-Agnt
Bounce Conductor-Baton Composer #-Agnt
Build Democratic-Nomination President #-Agnt
Build Note Black-Musicans #-Agnt
Burn P P #-Agnt
Capture Democratic-Nomination President #-Agnt
Control Animal Fruit #-Agnt
Control Apple-Core Apple #-Agnt
Control Apple-Core Tree #-Agnt
Control Bird Wings #-Agnt
Control Bob Story #-Ptnt
Control Bullet Gun #-Agnt
Control Bus Wheel #-Agnt
Control Camp-Fire Rock #-Agnt
Control Chair Wood #-Agnt
Control Chair-Leg Chair #-Agnt
Control Daddy-Bear Mommy-Bear #-Agnt
Control Eagle Eagle-Head #-Agnt
Control Field Team #-Agnt
Control Fish Water #-Agnt
Control Football Ball #-Agnt
Control Football Field #-Agnt
Control Footer Ground #-Agnt
Control Fortress Army #-Agnt
Control Furniture Human #-Agnt
Control Golf-Ball Golf-Green #-Agnt
Control Gun Barrell #-Agnt
Control Hammer-Head Hammer-Handle #-Agnt
Control Hammer-Head Human #-Agnt
Control Healthy-Tissue X-Ray #-Agnt
Control Horse Horse-Torso #-Agnt
Control Horse Human #-Agnt
Control House Brick #-Agnt
Control Insect Insect-Head #-Agnt
Control Insect Insect-Legs #-Agnt
Control Iron-Bar Coffee #-Agnt
Control Jfk White-House #-Agnt
Control John-Doe Car #-Agnt
Control Knife Blade #-Agnt
Control Knife Human #-Agnt
Control Lada Car #-Agnt
Control Orange-Core Apple #-Agnt
Control Orange-Core Orange #-Agnt
Control P P #-Agnt
Control Pig Pig-House #-Agnt
Control Roads Army #-Agnt
Control Rolls-Royce Car #-Agnt
Control Scissors Blade #-Agnt
Control Scissors Something #-Agnt
Control Seat Car #-Agnt
Control Shoe Foot #-Agnt
Control Swamp Army #-Agnt
Control Temperature-A Iron-Bar #-Agnt
Control Temperature-A Temperature-B #-Agnt
Control Veg-Substance Flower #-Agnt
Control Vehicle Wheel #-Agnt
Control Water Beaker #-Agnt
Control Weapon Person #-Agnt
Control Wings Eagle #-Agnt
Converge Bristle Italian-School #-Ptnt
Converge Line P #-Ptnt
Create Healthy-Tissue X-Ray #-Agnt
Create P Line #-Agnt
Cut Animal Fruit #-Agnt
Cut Apple-Core Apple #-Agnt
Cut Brain Man #-Agnt
Cut Bus Wheel #-Agnt
Cut Camp-Fire Rock #-Agnt
Cut Chair Wood #-Agnt
Cut Chair-Leg Chair-Legs #-Agnt
Cut Composer Listenership #-Agnt
Cut Daddy-Bear Mommy-Bear #-Agnt
Cut Eagle Eagle-Head #-Agnt
Cut Fish Water #-Agnt
Cut Football Field #-Agnt
Cut Footer Ground #-Agnt
Cut Furniture Human #-Agnt
Cut Golf-Ball Golf-Green #-Agnt
Cut Gun Barrell #-Agnt
Cut Hammer-Head Hammer-Handle #-Agnt
Cut Horse Human #-Agnt
Cut House Brick #-Agnt
Cut Insect Insect-Head #-Agnt
Cut Iron-Bar Coffee #-Agnt
Cut Jfk White-House #-Agnt
Cut Lada Car #-Agnt
Cut Lada Car #-Agnt
Cut Medical-Asistants Medical-Staff #-Agnt
Cut Orange-Core Orange #-Agnt
Cut P P #-Agnt
Cut Percussion Drum #-Agnt
Cut Pig Pig-House #-Agnt
Cut Planet Sun #-Agnt
Cut Real-Life Young #-Agnt
Cut Rolls-Royce Car #-Agnt
Cut Shoe Foot #-Agnt
Cut Temperature-A Temperature-B #-Agnt
Cut Vehicle Wheel #-Agnt
Cut Water Beaker #-Agnt
Cut Weapon Person #-Agnt
Damage Black-Musicans Violence #-Ptnt
Damage Listener Percussion #-Ptnt
Damage Real-Life Italian-School #-Agnt
Damage Tone Black-Musicans #-Agnt
Damage Violence Real-Life #-Agnt
Decorate Joe-Kennedy Democratic-Nomination #-Ptnt
Decorate P P #-Ptnt
Died Composer Musical-Instrument #-Agnt
Died P Line #-Agnt
Drive Animal Fruit #-Agnt
Drive Animal Fruit-Seed #-Agnt
Drive Apple-Core Apple #-Agnt
Drive Architect Blue-Print #-Ptnt
Drive Army Fortress #-Agnt
Drive Army Platoon #-Agnt
Drive Baby-Bear Mommy-Bear #-Agnt
Drive Bank Money #-Agnt

Drive Banker Money #-Ptnt
Drive Barrell Bullet #-Agnt
Drive Bird Wings #-Agnt
Drive Blade Something #-Agnt
Drive Blood Coffin #-Agnt
Drive Bob Football #-Ptnt
Drive Bob Rain #-Ptnt
Drive Bob Story #-Ptnt
Drive Brain Man #-Agnt
Drive Brick Sun #-Agnt
Drive Bus Wheel #-Agnt
Drive Camp-Fire Rock #-Agnt
Drive Candle Paper #-Agnt
Drive Chair Wood #-Agnt
Drive Chair-Leg Wood #-Agnt
Drive Clothes Human #-Agnt
Drive Coffee Temperature-B #-Agnt
Drive Composer Orchestra #-Agnt
Drive Conductor-Baton Listenership #-Agnt
Drive Daddy-Bear Mommy-Bear #-Agnt
Drive Democratic-Nomination Jfk #-Agnt
Drive Eagle Eagle-Head #-Agnt
Drive Fin Water #-Agnt
Drive Fish Water #-Agnt
Drive Flag Golf-Green #-Agnt
Drive Flesh Human #-Agnt
Drive Flower Stem #-Agnt
Drive Flower-Bloom Stem #-Agnt
Drive Football Field #-Agnt
Drive Footer Ground #-Agnt
Drive Furniture Human #-Agnt
Drive General Army #-Ptnt
Drive Golf-Ball Golf-Green #-Agnt
Drive Gun Barrell #-Agnt
Drive Habitation Planet #-Agnt
Drive Hammer-Head Hammer-Handle #-Agnt
Drive Healthy-Tissue Beam #-Agnt
Drive Horse Human #-Agnt
Drive House Brick #-Agnt
Drive House Human #-Agnt
Drive Human Hammer-Handle #-Ptnt
Drive Insect Insect-Head #-Agnt
Drive Insect-Body Insect-Head #-Agnt
Drive Iron-Bar Coffee #-Agnt
Drive Italian-School Paintbrush #-Agnt
Drive Jfk White-House #-Agnt
Drive John Sun #-Ptnt
Drive Knife Blade #-Agnt
Drive Knife-Handle Blade #-Agnt
Drive Lada Car #-Agnt
Drive Mary Sun #-Ptnt
Drive Mind Man #-Agnt
Drive Note Guitar #-Agnt
Drive Nucleus Electron #-Agnt
Drive Object Tool #-Agnt
Drive Orange-Core Orange #-Agnt
Drive P P #-Agnt
Drive Pig Pig-House #-Agnt
Drive Planet Sun #-Agnt
Drive Player Field #-Ptnt
Drive Roads Fortress #-Agnt
Drive Rolls-Royce Car #-Agnt
Drive Scissors Something #-Agnt
Drive Seat Wheel #-Agnt
Drive Shoe Foot #-Agnt
Drive Sick-People Operating-Procedure #-Ptnt
Drive Sky Wings #-Agnt
Drive Sun Earth #-Agnt
Drive Sun Snow #-Agnt
Drive Swamp Platoon #-Agnt
Drive Sword Society #-Agnt
Drive Temperature-A Coffee #-Agnt
Drive Temperature-A Temperature-B #-Agnt
Drive Tom Football #-Ptnt
Drive Tom House #-Ptnt
Drive Vampire Blood #-Ptnt
Drive Veg-Substance Apple #-Agnt
Drive Veg-Substance Orange #-Agnt
Drive Vehicle Wheel #-Agnt
Drive Vehicle-Body Wheel #-Agnt
Drive Vial Beaker #-Agnt
Drive Wall Ground #-Agnt
Drive Water Beaker #-Agnt
Drive Weapon Person #-Agnt
Drive X-Ray Beam #-Agnt
Eat Architect Blue-Print #-Ptnt
Eat Brain Mind #-Agnt
Eat Brain Mind #-Agnt
Eat Caravaggio Italian-School #-Ptnt
Eat Cpu Program #-Agnt
Eat Joe-Kennedy Jfk #-Ptnt
Eat P P #-Agnt
Execute Animal Fruit #-Agnt
Execute Apple-Core Veg-Substance #-Agnt
Execute Architect Blue-Print #-Agnt
Execute Bob Football #-Agnt
Execute Bob Rain #-Agnt
Execute Camp-Fire Rock #-Agnt
Execute Daddy-Bear Mommy-Bear #-Agnt
Execute Football Field #-Agnt
Execute Fortress Swamp #-Agnt
Execute Hammer-Head Hammer-Handle #-Agnt
Execute Healthy-Tissue Path #-Agnt
Execute Insect Insect-Body #-Agnt
Execute Jfk White-House #-Agnt
Execute Joe-Kennedy Democratic-Nomination #-Agnt
Execute John-Doe Car #-Agnt
Execute Lada Car #-Agnt
Execute Orange-Core Veg-Substance #-Agnt
Execute Pig Pig-House #-Agnt
Execute Planet Sun #-Agnt
Execute Roads Fortress #-Agnt
Execute Rolls-Royce Car #-Agnt
Execute Scissors Something #-Agnt
Execute Seat Car #-Agnt
Execute Swamp Path #-Agnt
Execute Temperature-A Temperature-B #-Agnt
Execute Tom Football #-Agnt
Execute Tom Gun #-Agnt
Execute Tom House #-Agnt
Execute Tom Mary #-Agnt
Execute Weapon Person #-Agnt
Go-Down Bristle Violence #-Agnt
Go-Down Note Tone #-Agnt
Go-Down Something Knife-Handle #-Agnt
Has-Part Democratic-Nomination Jfk #-Agnt
Has-Part Healthy-Tissue X-Ray #-Ptnt
Has-Part Joe-Kennedy Democratic-Nomination #-Ptnt
Has-Part Mind Man #-Agnt
Has-Part P P #-Agnt
Has-Part Story Bob #-Agnt
Has-Part Violence Paintbrush #-Agnt
Heavier Beam Tumour #-Agnt
Heavier Bob Story #-Agnt
Heavier Caravaggio Paintbrush #-Agnt
Heavier John Sun #-Agnt
Heavier P P #-Agnt
Heavier Paintbrush Caravaggio #-Agnt
Heavier Program Cpu #-Agnt
Heavier Story Bob #-Agnt
Heavier Tom Football #-Agnt
Holds Bob Story #-Ptnt
Holds X-Ray Healthy-Tissue #-Agnt
Inhabits Apple Orange-Core #-Ptnt
Inhabits Engine Man #-Agnt
Inhabits Football Ball #-Agnt
Inhabits Footer Wall #-Agnt
Inhabits Golf-Ball Flag #-Agnt
Inhabits Human Shoe-Sole #-Ptnt
Inhabits Insect Insect-Body #-Ptnt

Inhabits Tom Joe #-Ptnt
Inhabits X-Ray Healthy-Tissue #-Agnt
Inside Brain Man #-Ptnt
Inside Guitar Note #-Ptnt
Inside Healthy-Tissue Tumour #-Ptnt
Inside Leadbelly Rythm #-Ptnt
Inside Man Mind #-Ptnt
Inside P P #-Agnt
Inside Rythm Music #-Agnt
Inside Tom Mary #-Ptnt
Jealous-Of Animal Fruit-Seed #-Agnt
Jealous-Of Apple-Core Veg-Substance #-Agnt
Jealous-Of Daddy-Bear Baby-Bear #-Agnt
Jealous-Of Hammer-Head Human #-Agnt
Jealous-Of Human Object #-Ptnt
Jealous-Of Insect Insect-Body #-Agnt
Jealous-Of Orange-Core Veg-Substance #-Agnt
Jealous-Of Roads Army #-Agnt
Jealous-Of Scissors Blade #-Agnt
Lifestyle P Line #-Agnt
Live-In Animal Fruit-Seed #-Agnt
Live-In Daddy-Bear Baby-Bear #-Agnt
Live-In Engine Man #-Agnt
Live-In Football Player #-Agnt
Live-In Fortress Road #-Agnt
Live-In Hammer-Head Human #-Agnt
Live-In Healthy-Tissue Path #-Agnt
Live-In Insect Insect-Body #-Agnt
Live-In Planet Habitation #-Agnt
Live-In Roads Army #-Agnt
Live-In Scissors Blade #-Agnt
Live-In Swamp Path #-Agnt
Live-In Temperature-A Coffee #-Agnt
Lives-In Animal Fruit-Seed #-Ptnt
Lives-In Apple-Core Veg-Substance #-Agnt
Lives-In Daddy-Bear Baby-Bear #-Ptnt
Lives-In Hammer-Head Human #-Agnt
Lives-In Insect Insect-Body #-Ptnt
Lives-In Orange-Core Veg-Substance #-Agnt
Lives-In Roads Army #-Agnt
Lives-In Scissors Blade #-Agnt
Lives-In Tom Joe #-Ptnt
Lusts-After P P #-Agnt
Made-Of Joe-Kennedy Democratic-Nomination #-Ptnt
Made-Of Man Mind #-Ptnt
Make Healthy-Tissue Tumour #-Agnt
Make P P #-Agnt
Melt Man Mind #-Agnt
Obtain Blue-Print Builder #-Agnt
Obtain Guitar Note #-Agnt
On-Top-Of Caravaggio Paintbrush #-Agnt
On-Top-Of Composer Listenership #-Agnt
On-Top-Of John Sun #-Agnt
On-Top-Of P P #-Agnt
On-Top-Of Tom Mary #-Agnt
Owns Animal Fruit #-Agnt
Owns Bird Sky #-Agnt
Owns Brain Man #-Agnt
Owns Bus Human #-Agnt
Owns Camp-Fire Rock #-Agnt
Owns Candle Paper #-Agnt
Owns Chair Wood #-Agnt
Owns Chair Wood #-Agnt
Owns Clothes Human #-Agnt
Owns Composer Listenership #-Agnt
Owns Daddy-Bear Mommy-Bear #-Agnt
Owns Eagle Flesh #-Agnt
Owns Engine Car #-Agnt
Owns Footer Ground #-Agnt
Owns Fortress Swamp #-Agnt
Owns Furniture Human #-Agnt
Owns Golf-Ball Golf-Green #-Agnt
Owns Gun Metal #-Agnt
Owns Hammer-Head Hammer-Handle #-Agnt
Owns Healthy-Tissue Tumour #-Agnt
Owns Horse Human #-Agnt
Owns House Brick #-Agnt
Owns Jfk White-House #-Agnt
Owns John-Doe Car #-Agnt
Owns Knife Wood #-Agnt
Owns Lada Car #-Agnt
Owns Nucleus Electron #-Agnt
Owns Orange-Core Orange #-Agnt
Owns P P #-Agnt
Owns Pig Pig-House #-Agnt
Owns Planet Sun #-Agnt
Owns Roads Fortress #-Agnt
Owns Rolls-Royce Car A #-Agnt
Owns Scissors Something #-Agnt
Owns Scissors Something #-Agnt
Owns Seat Car #-Agnt
Owns Shoe Shoe-Sole #-Agnt
Owns Sun Earth #-Agnt
Owns Swamp Fortress #-Agnt
Owns Temperature-A Coffee #-Agnt
Owns Temperature-A Temperature-B #-Agnt
Owns Vehicle Human #-Agnt
Owns Water Beaker #-Agnt
Owns Weapon Person A #-Agnt
Paddle Apple-Core Apple #-Agnt
Paddle Architect Blue-Print #-Ptnt
Paddle Banker Money #-Ptnt
Paddle Bird Sky #-Agnt
Paddle Bob Football #-Ptnt
Paddle Bob Rain #-Ptnt
Paddle Bob Story #-Ptnt
Paddle Brain Man #-Agnt
Paddle Bus Wheel #-Agnt
Paddle Camp-Fire Rock #-Agnt
Paddle Candle Paper #-Agnt
Paddle Chair Wood #-Agnt
Paddle Clothes Human #-Agnt
Paddle Composer Listenership #-Agnt
Paddle Cpu Program #-Agnt
Paddle Daddy-Bear Mommy-Bear #-Agnt
Paddle Eagle Eagle-Head #-Agnt
Paddle Engine Car #-Agnt
Paddle Fish Water #-Agnt
Paddle Flower Stem #-Agnt
Paddle Football Field #-Agnt
Paddle Footer Ground #-Agnt
Paddle Fortress Swamp #-Agnt
Paddle Fruit Fruit-Seed #-Agnt
Paddle Furniture Human #-Agnt
Paddle General Society #-Ptnt
Paddle Golf-Ball Golf-Green #-Agnt
Paddle Gun Bullet #-Agnt
Paddle Hammer-Head Hammer-Handle #-Agnt
Paddle Healthy-Tissue Tumour #-Agnt
Paddle Horse Human #-Agnt
Paddle House Brick #-Agnt
Paddle Insect Insect-Head #-Agnt
Paddle Iron-Bar Coffee #-Agnt
Paddle Jfk White-House #-Agnt
Paddle John Shop #-Ptnt
Paddle John Sun #-Ptnt
Paddle Knife Blade #-Agnt
Paddle Lada Car #-Agnt
Paddle Lada Car #-Agnt
Paddle Mary Sun #-Ptnt
Paddle Nucleus Electron #-Agnt
Paddle Orange-Core Orange #-Agnt
Paddle P P #-Agnt
Paddle Pig Pig-House #-Agnt
Paddle Planet Sun #-Agnt
Paddle Roads Fortress #-Agnt
Paddle Rolls-Royce Car #-Agnt
Paddle Scissors Something #-Agnt
Paddle Seat Car #-Agnt
Paddle Shoe Foot #-Agnt

Paddle Sun Earth #-Agnt
Paddle Surgeon Operating-Procedure #-Ptnt
Paddle Swamp Fortress #-Agnt
Paddle Temperature-A Temperature-B #-Agnt
Paddle Tom Football #-Ptnt
Paddle Tom House #-Ptnt
Paddle Vampire Blood #-Ptnt
Paddle Vehicle Wheel #-Agnt
Paddle Water Beaker #-Agnt
Paddle Weapon Person #-Agnt
Propel Paintbrush Violence #-Agnt
Revolves Beam X-Ray #-Ptnt
Revolves P P #-Agnt
See Beam X-Ray #-Agnt
See Joe-Kennedy Democratic-Nomination #-Ptnt
Sit-In Healthy-Tissue X-Ray #-Agnt
Sit-In P P #-Agnt
Sit-In Painting Paintbrush #-Agnt
Sit-In Program Cpu #-Agnt
Sit-On Paintbrush Theatrical #-Agnt
Style Clubs Field #-Ptnt
Style Human Chair-Legs #-Ptnt
Style Knife-Handle Something #-Ptnt
Style Operating-Procedure Medical-Assistants #-Ptnt
Surround Democratic-Nomination Jfk #-Agnt
Tell Animal Fruit #-Ptnt
Tell Apple-Core Apple #-Ptnt
Tell Banker Money #-Ptnt
Tell Bob Football #-Ptnt
Tell Bob Rain #-Ptnt
Tell Brain Man .. PRED ADAPT,no! #-Ptnt
Tell Bus Wheel #-Ptnt
Tell Caravaggio Paintbrush #-Ptnt
Tell Chair Wood #-Ptnt
Tell Composer Listenership #-Ptnt
Tell Daddy-Bear Mommy-Bear #-Ptnt
Tell Eagle Eagle-Head #-Ptnt
Tell Fish Water #-Ptnt
Tell Furniture Human #-Ptnt
Tell General Society #-Ptnt
Tell Gun Barrell #-Ptnt
Tell Hammer-Head Hammer-Handle #-Ptnt
Tell Horse Human #-Ptnt
Tell House Brick #-Ptnt
Tell Insect Insect-Head #-Ptnt
Tell Iron-Bar Coffee #-Ptnt
Tell John Sun #-Ptnt
Tell Knife Blade #-Ptnt
Tell Lada Car #-Ptnt
Tell Orange-Core Orange #-Ptnt
Tell Pig Pig-House #-Ptnt
Tell Rolls-Royce Car #-Ptnt
Tell Scissors Something #-Ptnt
Tell Shoe Foot #-Ptnt
Tell Surgeon Operating-Procedure #-Ptnt
Tell Temperature-A Temperature-B #-Ptnt
Tell Tom Football #-Ptnt
Tell Tom Gun #-Ptnt
Tell Tom House #-Ptnt
Tell Tom Mary #-Ptnt
Tell Vehicle Wheel #-Ptnt
Tell Water Beaker #-Ptnt
Tell Weapon Person #-Ptnt
Throw P P #-Ptnt
Transport Brain Man #-Agnt
Transport Cpu Program #-Agnt
Transport P P #-Agnt
Transport Paintbrush Painting #-Agnt
Transport X-Ray Healthy-Tissue #-Agnt
Type-Of Theatrical Non-Derivitive #-Ptnt
Use Democratic-Nomination President #-Agnt
Use P Line #-Agnt
Use Percussion Composer #-Agnt
Use Real-Life Caravaggio #-Agnt
Use Violence Leadbelly #-Agnt
Works-In Animal Fruit-Seed #-Agnt
Works-In Apple-Core Veg-Substance #-Agnt
Works-In Daddy-Bear Baby-Bear #-Agnt
Works-In Engine Man #-Agnt
Works-In Football Player #-Agnt
Works-In Hammer-Head Human #-Agnt
Works-In Healthy-Tissue Path #-Agnt
Works-In Insect Insect-Body #-Agnt
Works-In Orange-Core Veg-Substance #-Agnt
Works-In Planet Habitation #-Agnt
Works-In Roads Army #-Agnt
Works-In Scissors Blade #-Agnt
Works-In Swamp Path #-Agnt
Works-In Temperature-A Coffee #-Agnt

Appendix D3

This is a list of the adapted inferences arising from the Assorted KB. The original inference and the adaptation are listed.

Lusts-After Animal Fruit #-Agnt	Become Animal Fruit
Affect Animal Fruit-Seed #-Agnt	Become Animal Fruit-Seed
Avoid Apple Orange #-Agnt	Become Apple Orange
Eat Apple-Core Apple #-Agnt	Become Apple-Core Apple
Influence Apple-Core Tree #-Agnt	Become Apple-Core Tree
Affect Apple-Core Veg-Substance #-Agnt	Become Apple-Core Veg-Substance
Go-Down Army Sword #-Agnt	Become Army Sword
Eat Banker Bank #-Ptnt	Become Banker Bank
Melt Barrell Metal #-Ptnt	Become Barrell Metal
Transport Blade Blade #-Agnt	Become Blade Blade
Avoid Blade Something #-Agnt	Become Blade Something
Melt Blade Wood #-Agnt	Become Blade Wood
Melt Brick Human #-Agnt	Become Brick Human
Go-Down Bus Seat #-Ptnt	Become Bus Seat
Lusts-After Bus Wheel #-Agnt	Become Bus Wheel
Found Coffee Temperature-A #-Ptnt	Become Coffee Temperature-A
Avoid Coffee Temperature-B #-Agnt	Become Coffee Temperature-B
Lusts-After Composer Listenership #-Agnt	Become Composer Listenership
Eat Composer Orchestra #-Agnt	Become Composer Orchestra
Lusts-After Daddy-Bear Mommy-Bear #-Agnt	Become Daddy-Bear Mommy-Bear
Found Eagle Eagle-Head #-Ptnt	Become Eagle Eagle-Head
Go-Down Eagle Eagle-Torso #-Ptnt	Become Eagle Eagle-Torso
Melt Eagle-Head Flesh #-Agnt	Become Eagle-Head Flesh
Build Excalibur King #-Agnt	Become Excalibur King
Eat Fish Fin #-Ptnt	Become Fish Fin
Lusts-After Fish Water #-Agnt	Become Fish Water
Go-Down Flower Flower-Bloom #-Agnt	Become Flower Flower-Bloom
Build Flower-Bloom Veg-Substance #-Agnt	Become Flower-Bloom Veg-Substance
Found Foot Shoe-Sole #-Ptnt	Become Foot Shoe-Sole
Make Fortress Swamp #-Agnt	Become Fortress Swamp
Lusts-After Furniture Human #-Agnt	Become Furniture Human
Eat General Army #-Ptnt	Become General Army
Eat General Society #-Ptnt	Become General Society
Found Hammer-Handle Hammer #-Ptnt	Become Hammer-Handle Hammer
Melt Hammer-Handle Human #-Agnt	Become Hammer-Handle Human
Lusts-After Hammer-Head Hammer-Handle #-Agnt	Become Hammer-Head Hammer-Handle
Buy Hammer-Head Human #-Agnt	Become Hammer-Head Human
Make Horse Human #-Agnt	Become Horse Human
Make House Brick #-Agnt	Become House Brick
Eat House Human #-Agnt	Become House Human
Melt Human Flesh #-Agnt	Become Human Flesh
Melt Human House #-Agnt	Become Human House
Go-Down Human Knife-Handle #-Ptnt	Become Human Knife-Handle
Capture Human Seat #-Agnt	Become Human Seat
Go-Down Human Shoe-Sole #-Ptnt	Become Human Shoe-Sole
Go-Down Human Vehicle-Body #-Ptnt	Become Human Vehicle-Body
Transport Insect Insect-Body #-Agnt	Become Insect Insect-Body
Lusts-After Insect Insect-Head #-Agnt	Become Insect Insect-Head
Use Insect Insect-Legs #-Agnt	Become Insect Insect-Legs
Capture Insect-Body Insect-Legs #-Agnt	Become Insect-Body Insect-Legs
Eat Jfk White-House #-Agnt	Become Jfk White-House
Eat John Brick #-Ptnt	Become John Brick
Lusts-After John-Doe Car #-Agnt	Become John-Doe Car
Lusts-After Knife Blade #-Agnt	Become Knife Blade
Transport Knife Wood #-Agnt	Become Knife Wood
Obtain Listenership Conductor-Baton #-Agnt	Become Listenership Conductor-Baton
Found Listenership Orchestra #-Ptnt	Become Listenership Orchestra
Eat Mary Sun #-Ptnt	Become Mary Sun
Eat Merlin Arthur #-Ptnt	Become Merlin Arthur
Go-Down Merlin King #-Ptnt	Become Merlin King
Melt Money Bank #-Ptnt	Become Money Bank
Go-Down Musician Conductor-Baton #-Ptnt	Become Musician Conductor-Baton
Lusts-After Nucleus Electron #-Agnt	Become Nucleus Electron

Found Operating-Procedure Patient #-Ptnt	Become Operating-Procedure Patient
Melt Operating-Procedure Sick-People #-Agnt	Become Operating-Procedure Sick-People
Make Orange-Core Orange #-Agnt	Become Orange-Core Orange
Go-Down Orchestra Conductor-Baton #-Agnt	Become Orchestra Conductor-Baton
Avoid Paintbrush Caravaggio #-Agnt	Become Paintbrush Caravaggio
Make Pig Pig-House #-Agnt	Become Pig Pig-House
Transport Roads Army #-Agnt	Become Roads Army
Lusts-After Roads Fortress #-Agnt	Become Roads Fortress
Eat Rolls-Royce Car #-Agnt	Become Rolls-Royce Car
Lusts-After Scissors Something #-Agnt	Become Scissors Something
Make Seat Car #-Agnt	Become Seat Car
Build Seat Human #-Agnt	Become Seat Human
Obtain Snow Sun #-Agnt	Become Snow Sun
Found Society Army #-Ptnt	Become Society Army
Obtain Society Sword #-Agnt	Become Society Sword
Go-Down Soldier Sword #-Ptnt	Become Soldier Sword
Melt Something Blade #-Ptnt	Become Something Blade
Go-Down Something Knife-Handle #-Agnt	Become Something Knife-Handle
Found Something Scisssors #-Ptnt	Become Something Scisssors
Go-Down Stem Flower-Bloom #-Agnt	Become Stem Flower-Bloom
Melt Stem Veg-Substance #-Ptnt	Become Stem Veg-Substance
Melt Sun Brick #-Ptnt	Become Sun Brick
See Sun Sun #-Agnt	Become Sun Sun
Eat Surgeon Operating-Procedure #-Ptnt	Become Surgeon Operating-Procedure
Make Swamp Fortress #-Agnt	Become Swamp Fortress
Found Swamp Platoon #-Ptnt	Become Swamp Platoon
Avoid Temperature-A Coffee #-Agnt	Become Temperature-A Coffee
Transport Temperature-A Ice-Cube #-Agnt	Become Temperature-A Ice-Cube
Make Temperature-A Temperature-B #-Agnt	Become Temperature-A Temperature-B
Found Temperature-B Ice-Cube #-Ptnt	Become Temperature-B Ice-Cube
Eat Tom Mary #-Ptnt	Become Tom Mary
Melt Tool Object #-Agnt	Become Tool Object
Go-Down Tree Veg-Substance #-Agnt	Become Tree Veg-Substance
Found Tumour Beam #-Ptnt	Become Tumour Beam
Eat Veg-Substance Apple-Core #-Agnt	Become Veg-Substance Apple-Core
Create Veg-Substance Flower-Bloom #-Agnt	Become Veg-Substance Flower-Bloom
Eat Veg-Substance Orange-Core #-Agnt	Become Veg-Substance Orange-Core
Capture Veg-Substance Tree #-Agnt	Become Veg-Substance Tree
Found Vehicle Vehicle-Body #-Ptnt	Become Vehicle Vehicle-Body
Lusts-After Vehicle Wheel #-Agnt	Become Vehicle Wheel
Capture Vehicle-Body Human #-Agnt	Become Vehicle-Body Human
Melt Water Fin #-Ptnt	Become Water Fin
Create Water Vial #-Agnt	Become Water Vial
Make Weapon Person #-Agnt	Become Weapon Person
Found Wheel Human #-Ptnt	Become Wheel Human
Obtain Wheel Seat #-Agnt	Become Wheel Seat
Obtain Wheel Vehicle-Body #-Agnt	Become Wheel Vehicle-Body
Melt Wood Chair-Seat #-Ptnt	Become Wood Chair-Seat
Heavier Bird Wings #-Agnt	Bigger-Than Bird Wings
Heavier Blade Knife #-Agnt	Bigger-Than Blade Knife
Heavier Brick House #-Agnt	Bigger-Than Brick House
Heavier Bus Wheel #-Agnt	Bigger-Than Bus Wheel
Heavier Car John-Doe #-Agnt	Bigger-Than Car John-Doe
Heavier Car Rolls-Royce #-Agnt	Bigger-Than Car Rolls-Royce
Heavier Composer Listenership #-Agnt	Bigger-Than Composer Listenership
Heavier Fish Water #-Agnt	Bigger-Than Fish Water
Heavier Footer Ground #-Agnt	Bigger-Than Footer Ground
Heavier General Society #-Agnt	Bigger-Than General Society
Heavier Knife Blade #-Agnt	Bigger-Than Knife Blade
Heavier Listenership Composer #-Agnt	Bigger-Than Listenership Composer
Heavier Mary Snow #-Agnt	Bigger-Than Mary Snow
Heavier Money Banker #-Agnt	Bigger-Than Money Banker
Heavier Person Weapon #-Agnt	Bigger-Than Person Weapon
Heavier Pig Pig-House #-Agnt	Bigger-Than Pig Pig-House
Heavier Pig-House Pig #-Agnt	Bigger-Than Pig-House Pig
Heavier Platoon Fortress #-Agnt	Bigger-Than Platoon Fortress
Heavier Rolls-Royce Car #-Agnt	Bigger-Than Rolls-Royce Car
Heavier Sky Bird #-Agnt	Bigger-Than Sky Bird
Heavier Society General #-Agnt	Bigger-Than Society General
Heavier Vampire Coffin #-Agnt	Bigger-Than Vampire Coffin
Heavier Water Fish #-Agnt	Bigger-Than Water Fish
Heavier Weapon Person #-Agnt	Bigger-Than Weapon Person
Heavier Wheel Bus #-Agnt	Bigger-Than Wheel Bus
Melt Mary Joe #-Agnt	Burn Mary Joe

Eat Leadbelly Black-Musicans #-Ptnt	Control Leadbelly Black-Musicans
Paddle John-Doe Car #-Agnt	Cycle John-Doe Car
On-Top-Of Animal Fruit #-Agnt	Far-From Animal Fruit
On-Top-Of Apple-Core Apple #-Agnt	Far-From Apple-Core Apple
On-Top-Of Banker Money #-Agnt	Far-From Banker Money
On-Top-Of Bob Football #-Agnt	Far-From Bob Football
On-Top-Of Bob Rain #-Agnt	Far-From Bob Rain
On-Top-Of Brain Man #-Agnt	Far-From Brain Man
On-Top-Of Bus Wheel #-Agnt	Far-From Bus Wheel
On-Top-Of Camp-Fire Rock #-Agnt	Far-From Camp-Fire Rock
On-Top-Of Chair Wood #-Agnt	Far-From Chair Wood
On-Top-Of Daddy-Bear Mommy-Bear #-Agnt	Far-From Daddy-Bear Mommy-Bear
On-Top-Of Eagle Eagle-Head #-Agnt	Far-From Eagle Eagle-Head
On-Top-Of Fish Water #-Agnt	Far-From Fish Water
On-Top-Of Furniture Human #-Agnt	Far-From Furniture Human
On-Top-Of General Society #-Agnt	Far-From General Society
On-Top-Of Gun Bullet #-Agnt	Far-From Gun Bullet
On-Top-Of Hammer-Head Hammer-Handle #-Agnt	Far-From Hammer-Head Hammer-Handle
On-Top-Of Horse Human #-Agnt	Far-From Horse Human
On-Top-Of House Brick #-Agnt	Far-From House Brick
On-Top-Of Insect Insect-Head #-Agnt	Far-From Insect Insect-Head
On-Top-Of Iron-Bar Coffee #-Agnt	Far-From Iron-Bar Coffee
On-Top-Of Jfk White-House #-Agnt	Far-From Jfk White-House
On-Top-Of John-Doe Car #-Agnt	Far-From John-Doe Car
On-Top-Of Knife Blade #-Agnt	Far-From Knife Blade
On-Top-Of Lada Car #-Agnt	Far-From Lada Car
On-Top-Of Orange-Core Orange #-Agnt	Far-From Orange-Core Orange
On-Top-Of Pig Pig-House #-Agnt	Far-From Pig Pig-House
On-Top-Of Rolls-Royce Car #-Agnt	Far-From Rolls-Royce Car
On-Top-Of Scissors Something #-Agnt	Far-From Scissors Something
On-Top-Of Seat Car #-Agnt	Far-From Seat Car
On-Top-Of Shoe Foot #-Agnt	Far-From Shoe Foot
On-Top-Of Surgeon Operating-Procedure #-Agnt	Far-From Surgeon Operating-Procedure
On-Top-Of Temperature-A Temperature-B #-Agnt	Far-From Temperature-A Temperature-B
On-Top-Of Tom Football #-Agnt	Far-From Tom Football
On-Top-Of Tom Gun #-Agnt	Far-From Tom Gun
On-Top-Of Tom House #-Agnt	Far-From Tom House
On-Top-Of Vehicle Wheel #-Agnt	Far-From Vehicle Wheel
On-Top-Of Water Beaker #-Ptnt	Far-From Water Beaker
On-Top-Of Weapon Person #-Agnt	Far-From Weapon Person
Eat Tom Gun #-Ptnt	Flies Tom Gun
Revolves Program Cpu #-Agnt	Flow Program Cpu
Inside Fish Water #-Ptnt	Found-In Fish Water
Inside Fortress Path #-Ptnt	Found-In Fortress Path
Inside Golf-Ball Golf-Green #-Ptnt	Found-In Golf-Ball Golf-Green
Inside Iron-Bar Coffee #-Ptnt	Found-In Iron-Bar Coffee
Inside Pig Pig-House #-Ptnt	Found-In Pig Pig-House
Inside Temperature-B Coffee #-Ptnt	Found-In Temperature-B Coffee
Inside Tumour Path #-Ptnt	Found-In Tumour Path
Flies-Through Apple Orange-Core #-Ptnt	Go-To Apple Orange-Core
Flies-Through Apple Veg-Substance #-Ptnt	Go-To Apple Veg-Substance
Flies-Through Banker Money #-Ptnt	Go-To Banker Money
Flies-Through Bob Football #-Ptnt	Go-To Bob Football
Flies-Through Bob Rain #-Ptnt	Go-To Bob Rain
Flies-Through Brain Man #-Agnt	Go-To Brain Man
Flies-Through Bus Wheel #-Agnt	Go-To Bus Wheel
Flies-Through Camp-Fire Rock #-Agnt	Go-To Camp-Fire Rock
Flies-Through Caravaggio Paintbrush #-Ptnt	Go-To Caravaggio Paintbrush
Flies-Through Chair Wood #-Agnt	Go-To Chair Wood
Flies-Through Clothes Human #-Agnt	Go-To Clothes Human
Flies-Through Composer Listenership #-Agnt	Go-To Composer Listenership
Flies-Through Daddy-Bear Mommy-Bear #-Ptnt	Go-To Daddy-Bear Mommy-Bear
Flies-Through Eagle Eagle-Head #-Ptnt	Go-To Eagle Eagle-Head
Flies-Through Fish Water #-Ptnt	Go-To Fish Water
Flies-Through Furniture Human #-Agnt	Go-To Furniture Human
Flies-Through General Society #-Ptnt	Go-To General Society
Flies-Through Gun Barrell #-Agnt	Go-To Gun Barrell
Flies-Through Hammer-Head Hammer-Handle #-Agnt	Go-To Hammer-Head Hammer-Handle
Flies-Through Horse Human #-Agnt	Go-To Horse Human
Flies-Through House Brick #-Agnt	Go-To House Brick
Flies-Through Human Shoe-Sole #-Ptnt	Go-To Human Shoe-Sole
Flies-Through Human Tool #-Ptnt	Go-To Human Tool
Flies-Through Insect Insect-Body #-Ptnt	Go-To Insect Insect-Body
Flies-Through Jfk White-House #-Agnt	Go-To Jfk White-House

Flies-Through John Sun #-Ptnt	Go-To John Sun
Flies-Through Knife Wood #-Agnt	Go-To Knife Wood
Flies-Through Lada Car #-Agnt	Go-To Lada Car
Flies-Through Lada Car #-Agnt	Go-To Lada Car
Flies-Through Nucleus Electron #-Agnt	Go-To Nucleus Electron
Flies-Through P P #-Agnt	Go-To P P
Flies-Through Pig Pig-House #-Ptnt	Go-To Pig Pig-House
Flies-Through Planet Sun #-Agnt	Go-To Planet Sun
Flies-Through Rolls-Royce Car #-Agnt	Go-To Rolls-Royce Car
Flies-Through Scissors Something #-Agnt	Go-To Scissors Something
Flies-Through Temperature-A Coffee #-Agnt	Go-To Temperature-A Coffee
Flies-Through Temperature-A Temperature-B #-Agnt	Go-To Temperature-A Temperature-B
Flies-Through Tom Football #-Ptnt	Go-To Tom Football
Flies-Through Tom Gun #-Ptnt	Go-To Tom Gun
Flies-Through Tom Mary #-Ptnt	Go-To Tom Mary
Flies-Through Water Beaker #-Agnt	Go-To Water Beaker
Flies-Through Weapon Person #-Agnt	Go-To Weapon Person
Heavier Mary Tom #-Agnt	Greater Mary Tom
Inhabits Human Shoe-Sole #-Ptnt	Hates Human Shoe-Sole
Inside Banker Money #-Ptnt	Holds Banker Money
Inside Bob Football #-Ptnt	Holds Bob Football
Inside Bob Rain #-Ptnt	Holds Bob Rain
Inside Caravaggio Paintbrush #-Ptnt	Holds Caravaggio Paintbrush
Inside General Society #-Ptnt	Holds General Society
Inside John Shop #-Ptnt	Holds John Shop
Inside John Sun #-Ptnt	Holds John Sun
Inside Mary Joe #-Ptnt	Holds Mary Joe
Inside Surgeon Operating-Procedure #-Ptnt	Holds Surgeon Operating-Procedure
Inside Tom Football #-Ptnt	Holds Tom Football
Inside Tom Gun #-Ptnt	Holds Tom Gun
Eat Tom House #-Ptnt	Holds Tom House
Eat Tom Joe #-Ptnt	Holds Tom Joe
Inside Tom Mary #-Ptnt	Holds Tom Mary
Make Animal Fruit #-Agnt	Injure Animal Fruit
Avoid Apple Orange-Core #-Agnt	Injure Apple Orange-Core
Go-Down Apple Veg-Substance #-Ptnt	Injure Apple Veg-Substance
Lusts-After Apple-Core Apple #-Agnt	Injure Apple-Core Apple
Transport Apple-Core Tree #-Agnt	Injure Apple-Core Tree
Buy Apple-Core Veg-Substance #-Agnt	Injure Apple-Core Veg-Substance
Propel Army Civilian #-Agnt	Injure Army Civilian
Eat Banker Money #-Ptnt	Injure Banker Money
Attack Barrell Bullet #-Agnt	Injure Barrell Bullet
Go-Down Barrell Handle #-Agnt	Injure Barrell Handle
See Beaker Water #-Agnt	Injure Beaker Water
Transport Bird Wings #-Agnt	Injure Bird Wings
Avoid Black-Musicans Leadbelly #-Agnt	Injure Black-Musicans Leadbelly
Go-Down Blade Blade #-Agnt	Injure Blade Blade
Avoid Blade Scissors #-Agnt	Injure Blade Scissors
Make Brain Man #-Agnt	Injure Brain Man
Attack Bristle Caravaggio #-Agnt	Injure Bristle Caravaggio
Affect Bullet Gun #-Agnt	Injure Bullet Gun
Make Camp-Fire Rock #-Agnt	Injure Camp-Fire Rock
Obtain Car Man #-Agnt	Injure Car Man
Eat Chair Chair-Seat #-Agnt	Injure Chair Chair-Seat
Go-Down Chair Wood #-Ptnt	Injure Chair Wood
Shoots-With Chair-Back Chair #-Agnt	Injure Chair-Back Chair
Use Chair-Leg Chair #-Agnt	Injure Chair-Leg Chair
Affect Chair-Leg Chair-Legs #-Agnt	Injure Chair-Leg Chair-Legs
Create Chair-Leg Wood #-Agnt	Injure Chair-Leg Wood
Attack Chair-Seat Chair-Leg #-Agnt	Injure Chair-Seat Chair-Leg
Go-Down Chair-Seat Wood #-Agnt	Injure Chair-Seat Wood
Lusts-After Clothes Human #-Agnt	Injure Clothes Human
Treatment-Of Clubs Studs #-Agnt	Injure Clubs Studs
Hit Daddy-Bear Baby-Bear #-Agnt	Injure Daddy-Bear Baby-Bear
Eat Daddy-Bear Mommy-Bear #-Ptnt	Injure Daddy-Bear Mommy-Bear
Lusts-After Eagle Eagle-Head #-Agnt	Injure Eagle Eagle-Head
Buy Eagle Eagle-Torso #-Agnt	Injure Eagle Eagle-Torso
Transport Eagle Flesh #-Agnt	Injure Eagle Flesh
Transport Eagle Wings #-Agnt	Injure Eagle Wings
Go-Down Eagle-Head Eagle-Torso #-Agnt	Injure Eagle-Head Eagle-Torso
Obtain Eagle-Head Flesh #-Agnt	Injure Eagle-Head Flesh
Attack Eagle-Head Wings #-Agnt	Injure Eagle-Head Wings
Affect Eagle-Torso Eagle #-Agnt	Injure Eagle-Torso Eagle
Build Eagle-Torso Flesh #-Agnt	Injure Eagle-Torso Flesh

Hoards Engine Man #-Agnt	Injure Engine Man
Affect Field Clubs #-Agnt	Injure Field Clubs
Obtain Field Player #-Agnt	Injure Field Player
Attack Field Team #-Agnt	Injure Field Team
Make Fish Water #-Agnt	Injure Fish Water
Capture Flesh Eagle-Torso #-Agnt	Injure Flesh Eagle-Torso
Transport Flower Flower-Bloom #-Agnt	Injure Flower Flower-Bloom
Transport Flower Veg-Substance #-Agnt	Injure Flower Veg-Substance
Capture Flower-Bloom Veg-Substance #-Agnt	Injure Flower-Bloom Veg-Substance
Hoards Football Ball #-Agnt	Injure Football Ball
Eat Football Field #-Agnt	Injure Football Field
Go-Down Football Goal-Keeper #-Agnt	Injure Football Goal-Keeper
Avoid Football Player #-Agnt	Injure Football Player
Affect Football Team #-Agnt	Injure Football Team
Eat Footer Ground #-Agnt	Injure Footer Ground
See Footer Wall #-Agnt	Injure Footer Wall
Affect Fortress Army #-Agnt	Injure Fortress Army
Propel Fruit Fruit-Seed #-Agnt	Injure Fruit Fruit-Seed
See Golf-Ball Flag #-Agnt	Injure Golf-Ball Flag
Eat Golf-Ball Golf-Green #-Agnt	Injure Golf-Ball Golf-Green
Lusts-After Gun Barrell #-Agnt	Injure Gun Barrell
Go-Down Gun Handle #-Ptnt	Injure Gun Handle
Transport Gun Metal #-Agnt	Injure Gun Metal
Avoid Gun Something #-Agnt	Injure Gun Something
Avoid Hammer Hammer-Handle #-Agnt	Injure Hammer Hammer-Handle
Go-Down Hammer Human #-Ptnt	Injure Hammer Human
Propel Hammer-Head Hammer #-Agnt	Injure Hammer-Head Hammer
Make Hammer-Head Hammer-Handle #-Agnt	Injure Hammer-Head Hammer-Handle
Hit Hammer-Head Human #-Agnt	Injure Hammer-Head Human
Influence Hammer-Head Wall #-Agnt	Injure Hammer-Head Wall
Eat Horse Flesh #-Agnt	Injure Horse Flesh
Lusts-After Horse Human #-Agnt	Injure Horse Human
Go-Down Horse-Head Flesh #-Agnt	Injure Horse-Head Flesh
Avoid Horse-Head Horse #-Agnt	Injure Horse-Head Horse
Avoid Horse-Head Human #-Agnt	Injure Horse-Head Human
Use Horse-Legs Horse #-Agnt	Injure Horse-Legs Horse
Go-Down Horse-Torso Flesh #-Agnt	Injure Horse-Torso Flesh
Capture Human Handle #-Agnt	Injure Human Handle
Eat Human Object #-Ptnt	Injure Human Object
Eat Human Shoe #-Ptnt	Injure Human Shoe
Capture Human Shoe-Sole #-Agnt	Injure Human Shoe-Sole
Buy Insect Insect-Body #-Agnt	Injure Insect Insect-Body
Eat Insect Insect-Head #-Ptnt	Injure Insect Insect-Head
Transport Insect Insect-Legs #-Agnt	Injure Insect Insect-Legs
Attack Insect-Legs Insect #-Agnt	Injure Insect-Legs Insect
Go-Down Insect-Legs Insect-Body #-Agnt	Injure Insect-Legs Insect-Body
Eat Iron-Bar Coffee #-Agnt	Injure Iron-Bar Coffee
Eat John Sun #-Ptnt	Injure John Sun
Create Knife Knife-Handle #-Agnt	Injure Knife Knife-Handle
Make Lada Car #-Agnt	Injure Lada Car
Make Lada Car #-Agnt	Injure Lada Car
Eat Man Engine #-Ptnt	Injure Man Engine
Use Medical-Asistants Surgeon #-Agnt	Injure Medical-Asistants Surgeon
Lusts-After Nucleus Electron #-Agnt	Injure Nucleus Electron
Obtain Operating-Procedure Sick-People #-Agnt	Injure Operating-Procedure Sick-People
Use Orange Apple #-Agnt	Injure Orange Apple
Buy Orange-Core Apple #-Agnt	Injure Orange-Core Apple
Lusts-After Orange-Core Orange #-Agnt	Injure Orange-Core Orange
Transport Orange-Core Tree #-Agnt	Injure Orange-Core Tree
Hit Orange-Core Veg-Substance #-Agnt	Injure Orange-Core Veg-Substance
Go-Down Orange-Peel Veg-Substance #-Agnt	Injure Orange-Peel Veg-Substance
Avoid Paintbrush Caravaggio #-Agnt	Injure Paintbrush Caravaggio
Eat Planet Habitation #-Agnt	Injure Planet Habitation
Eat Planet Sun #-Agnt	Injure Planet Sun
Buy Scissors Blade #-Agnt	Injure Scissors Blade
Go-Down Scisssors Blade #-Ptnt	Injure Scisssors Blade
Lusts-After Seat Car #-Agnt	Injure Seat Car
Bounce Seat Human #-Agnt	Injure Seat Human
Lusts-After Shoe Foot #-Agnt	Injure Shoe Foot
Buy Shoe Human #-Agnt	Injure Shoe Human
Transport Shoe Shoe-Sole #-Agnt	Injure Shoe Shoe-Sole
Go-Down Shoe-Sole Human #-Agnt	Injure Shoe-Sole Human
Attack Shoe-Upper Shoe #-Agnt	Injure Shoe-Upper Shoe
Attack Something Knife #-Agnt	Injure Something Knife

Obtain Stem Flower-Bloom #-Agnt	Injure Stem Flower-Bloom
Hoards Sun Habitation #-Agnt	Injure Sun Habitation
See Sun Sun #-Agnt	Injure Sun Sun
Affect Swamp Army #-Agnt	Injure Swamp Army
Bounce Sword General #-Agnt	Injure Sword General
Keep-Out Table Something #-Agnt	Injure Table Something
Propel Team Keeper #-Agnt	Injure Team Keeper
Buy Temperature-A Coffee #-Agnt	Injure Temperature-A Coffee
Influence Temperature-A Ice-Cube #-Agnt	Injure Temperature-A Ice-Cube
Eat Temperature-A Iron-Bar #-Agnt	Injure Temperature-A Iron-Bar
Eat Temperature-A Temperature-B #-Agnt	Injure Temperature-A Temperature-B
Obtain Temperature-B Coffee #-Agnt	Injure Temperature-B Coffee
Attack Temperature-B Heat #-Agnt	Injure Temperature-B Heat
Transport Temperature-B Iron-Bar #-Agnt	Injure Temperature-B Iron-Bar
Obtain Tool Object #-Agnt	Injure Tool Object
Attack Tree Apple-Core #-Agnt	Injure Tree Apple-Core
Obtain Tumour Healthy-Tissue #-Agnt	Injure Tumour Healthy-Tissue
Affect Veg-Substance Flower #-Agnt	Injure Veg-Substance Flower
Obtain Vial Water #-Agnt	Injure Vial Water
Eat Water Beaker #-Agnt	Injure Water Beaker
Eat Water Vial #-Agnt	Injure Water Vial
Shoots-With Wings Eagle #-Agnt	Injure Wings Eagle
Obtain Wood Chair-Leg #-Agnt	Injure Wood Chair-Leg
Sit-In Apple John #-Agnt	Lie-On Apple John
Sit-In Bank Banker #-Agnt	Lie-On Bank Banker
Sit-In Blade Blade #-Agnt	Lie-On Blade Blade
Surround Blue-Print House #-Agnt	Lie-On Blue-Print House
Sit-In Brick John #-Agnt	Lie-On Brick John
Sit-In Coffin Vampire #-Agnt	Lie-On Coffin Vampire
Sit-In Field Team #-Agnt	Lie-On Field Team
Sit-In Fin Fish #-Agnt	Lie-On Fin Fish
Sit-In Flag Golf-Ball #-Agnt	Lie-On Flag Golf-Ball
Sit-In Flesh Eagle #-Agnt	Lie-On Flesh Eagle
Sit-In Guitar Leadbelly #-Agnt	Lie-On Guitar Leadbelly
Sit-In House Architect #-Agnt	Lie-On House Architect
Sit-In House Furniture #-Agnt	Lie-On House Furniture
Sit-In Ice-Cube Temperature-A #-Agnt	Lie-On Ice-Cube Temperature-A
Sit-In Insect-Legs Insect #-Agnt	Lie-On Insect-Legs Insect
Sit-In Iron-Bar Temperature-B #-Agnt	Lie-On Iron-Bar Temperature-B
Sit-In Metal Gun #-Agnt	Lie-On Metal Gun
Sit-In Shoe-Sole Shoe #-Agnt	Lie-On Shoe-Sole Shoe
Sit-In Snow Mary #-Agnt	Lie-On Snow Mary
Sit-In Swamp Army #-Agnt	Lie-On Swamp Army
Sit-In Tree Apple-Core #-Agnt	Lie-On Tree Apple-Core
Sit-In Tree Orange-Core #-Agnt	Lie-On Tree Orange-Core
Sit-In Veg-Substance Flower #-Agnt	Lie-On Veg-Substance Flower
Sit-In Vial Water #-Agnt	Lie-On Vial Water
Sit-In Wall Footer #-Agnt	Lie-On Wall Footer
Sit-In Wings Bird #-Agnt	Lie-On Wings Bird
Sit-In Wood Chair #-Agnt	Lie-On Wood Chair
Sit-In Wood Knife #-Agnt	Lie-On Wood Knife
Connect Brain Mind #-Ptnt	Line Brain Mind
Connect Italian-School Bristle #-Agnt	Line Italian-School Bristle
Connect Man Mind #-Ptnt	Line Man Mind
Connect P P #-Agnt	Line P P
Connect Percussion Musician #-Agnt	Line Percussion Musician
Connect Violence Bristle #-Agnt	Line Violence Bristle
Connect X-Ray Path #-Agnt	Line X-Ray Path
Owns Fish Water #-Agnt	Lives-In Fish Water
Propel Architect Builder #-Agnt	Made-Of Architect Builder
Found Blue-Print House #-Ptnt	Made-Of Blue-Print House
Revolves Story Bob #-Agnt	Made-Of Story Bob
Inhabits Architect Builder #-Ptnt	Married-To Architect Builder
Inhabits Joe-KennedyPresident #-Ptnt	Married-To Joe-Kennedy President
Lives-In Tom Joe #-Ptnt	Married-To Tom Joe
Connect Music Note #-Ptnt	Merged-With Music Note
Connect Rythm Note #-Agnt	Merged-With Rythm Note
Connect Rythm Tone #-Agnt	Merged-With Rythm Tone
Connect Tone Note #-Agnt	Merged-With Tone Note
Connect Violence Note #-Agnt	Merged-With Violence Note
On-Top-Of Lada Car #-Agnt	Near Lada Car
Surround Excalibur Arthur #-Agnt	Next-To Excalibur Arthur
Surround Flesh Wings #-Agnt	Next-To Flesh Wings
Surround Habitation Planet #-Agnt	Next-To Habitation Planet

Surround Ice-Cube Temperature-B #-Agnt	Next-To Ice-Cube Temperature-B
Surround Iron-Bar Temperature-B #-Agnt	Next-To Iron-Bar Temperature-B
Surround Obj_Fortress Fortress #-Agnt	Next-To Obj_Fortress Fortress
Surround Planet Habitation #-Agnt	Next-To Planet Habitation
Surround Wings Flesh #-Agnt	Next-To Wings Flesh
Holds Animal Fruit #-Agnt	Outside Animal Fruit
Inside Architect Blue-Print #-Ptnt	Outside Architect Blue-Print
Inside Banker Money #-Ptnt	Outside Banker Money
Inside Barrell Handle #-Ptnt	Outside Barrell Handle
Inside Blade Knife #-Ptnt	Outside Blade Knife
Inside Blood Coffin #-Ptnt	Outside Blood Coffin
Inside Blue-Print Builder #-Ptnt	Outside Blue-Print Builder
Inside Brain Man #-Ptnt	Outside Brain Man
Inside Brick Human #-Ptnt	Outside Brick Human
Inside Bus Wheel #-Ptnt	Outside Bus Wheel
Inside Camp-Fire Rock #-Ptnt	Outside Camp-Fire Rock
Inside Caravaggio Paintbrush #-Ptnt	Outside Caravaggio Paintbrush
Inside Chair Wood #-Ptnt	Outside Chair Wood
Inside Chair-Back Human #-Ptnt	Outside Chair-Back Human
Surround Chair-Leg Human #-Agnt	Outside Chair-Leg Human
Inside Coffee Temperature-A #-Ptnt	Outside Coffee Temperature-A
Inside Composer Listenership #-Ptnt	Outside Composer Listenership
Inside Daddy-Bear Mommy-Bear #-Ptnt	Outside Daddy-Bear Mommy-Bear
Inside Eagle Eagle-Head #-Ptnt	Outside Eagle Eagle-Head
Inside Eagle-Head Flesh #-Ptnt	Outside Eagle-Head Flesh
Inside Eagle-Torso Eagle #-Ptnt	Outside Eagle-Torso Eagle
Inside Eagle-Torso Flesh #-Ptnt	Outside Eagle-Torso Flesh
Surround Excalibur Arthur #-Agnt	Outside Excalibur Arthur
Inside Field Player #-Ptnt	Outside Field Player
Inside Fin Fish #-Ptnt	Outside Fin Fish
Inside Fish Water #-Ptnt	Outside Fish Water
Surround Flag Golf-Green #-Agnt	Outside Flag Golf-Green
Surround Flesh Wings #-Agnt	Outside Flesh Wings
Inside Flower-Bloom Flower #-Ptnt	Outside Flower-Bloom Flower
Inside Football Field #-Ptnt	Outside Football Field
Inside Footer Ground #-Ptnt	Outside Footer Ground
Inside Fortress Army #-Ptnt	Outside Fortress Army
Found Fortress Platoon #-Ptnt	Outside Fortress Platoon
Inside Fortress Swamp #-Ptnt	Outside Fortress Swamp
Inside Furniture Human #-Ptnt	Outside Furniture Human
Inside General Society #-Ptnt	Outside General Society
Inside Golf-Green Flag #-Ptnt	Outside Golf-Green Flag
Inside Ground Wall #-Ptnt	Outside Ground Wall
Inside Gun Bullet #-Ptnt	Outside Gun Bullet
Inside Hammer-Handle Human #-Ptnt	Outside Hammer-Handle Human
Inside Hammer-Head Hammer #-Ptnt	Outside Hammer-Head Hammer
Inside Hammer-Head Hammer-Handle #-Ptnt	Outside Hammer-Head Hammer-Handle
Inside Horse Human #-Ptnt	Outside Horse Human
Inside Horse-Head Horse #-Ptnt	Outside Horse-Head Horse
Inside House Brick #-Ptnt	Outside House Brick
Inside Human Chair #-Ptnt	Outside Human Chair
Inside Human Flesh #-Ptnt	Outside Human Flesh
Inside Human Horse-Head #-Ptnt	Outside Human Horse-Head
Inside Human House #-Ptnt	Outside Human House
Inside Human Wheel #-Ptnt	Outside Human Wheel
Inside Insect Insect-Head #-Ptnt	Outside Insect Insect-Head
Inside Insect-Head Insect #-Ptnt	Outside Insect-Head Insect
Inside Insect-Legs Insect-Head #-Ptnt	Outside Insect-Legs Insect-Head
Inside Iron-Bar Coffee #-Ptnt	Outside Iron-Bar Coffee
Inside Jfk White-House #-Ptnt	Outside Jfk White-House
Inside Joe-Kennedy Jfk #-Ptnt	Outside Joe-Kennedy Jfk
Inside John Sun #-Ptnt	Outside John Sun
Holds John-Doe Car #-Agnt	Outside John-Doe Car
Inside Knife Blade #-Ptnt	Outside Knife Blade
Inside Lada Car #-Ptnt	Outside Lada Car
Inside Lada Car #-Ptnt	Outside Lada Car
Inside Listenership Conductor-Baton #-Ptnt	Outside Listenership Conductor-Baton
Inside Mary Joe #-Ptnt	Outside Mary Joe
Inside Mary Sun #-Ptnt	Outside Mary Sun
Inside Merlin Arthur #-Ptnt	Outside Merlin Arthur
Inside Mommy-Bear Baby-Bear #-Ptnt	Outside Mommy-Bear Baby-Bear
Inside Money Bank #-Ptnt	Outside Money Bank
Inside Operating-Procedure Sick-People #-Ptnt	Outside Operating-Procedure Sick-People
Inside Planet Sun #-Ptnt	Outside Planet Sun

Inside Roads Fortress #-Ptnt	Outside Roads Fortress
Inside Rolls-Royce Car #-Ptnt	Outside Rolls-Royce Car
Inside Scissors Something #-Ptnt	Outside Scissors Something
Holds Seat Car #-Agnt	Outside Seat Car
Inside Sky Wings #-Ptnt	Outside Sky Wings
Inside Society Sword #-Ptnt	Outside Society Sword
Inside Something Blade #-Ptnt	Outside Something Blade
Inside Stem Flower-Bloom #-Ptnt	Outside Stem Flower-Bloom
Inside Sun Brick #-Ptnt	Outside Sun Brick
Inside Sun Habitation #-Ptnt	Outside Sun Habitation
Inside Surgeon Operating-Procedure #-Ptnt	Outside Surgeon Operating-Procedure
Inside Swamp Fortress #-Ptnt	Outside Swamp Fortress
Inside Swamp Road #-Ptnt	Outside Swamp Road
Inside Temperature-A Temperature-B #-Ptnt	Outside Temperature-A Temperature-B
Inside Tom Football #-Ptnt	Outside Tom Football
Inside Tom Mary #-Ptnt	Outside Tom Mary
Inside Tool Object #-Ptnt	Outside Tool Object
Inside Vehicle Wheel #-Ptnt	Outside Vehicle Wheel
Inside Water Beaker #-Ptnt	Outside Water Beaker
Inside Water Fin #-Ptnt	Outside Water Fin
Inside Weapon Person #-Ptnt	Outside Weapon Person
Inside Wheel Seat #-Ptnt	Outside Wheel Seat
Inside Wheel Vehicle #-Ptnt	Outside Wheel Vehicle
Inside Wheel Vehicle-Body #-Ptnt	Outside Wheel Vehicle-Body
Inside Wings Bird #-Ptnt	Outside Wings Bird
Inside Wood Blade #-Ptnt	Outside Wood Blade
Inside Wood Chair-Leg #-Ptnt	Outside Wood Chair-Leg
Eat Bob Football #-Ptnt	Played-With Bob Football
Eat Bob Rain #-Ptnt	Played-With Bob Rain
Go-Down Patient Medical-Asistants #-Ptnt	Played-With Patient Medical-Asistants
Propel Sick-People Instruments #-Agnt	Played-With Sick-People Instruments
Go-Down Sick-People Medical-Asistants #-Ptnt	Played-With Sick-People Medical-Asistants
Eat Surgeon Sick-People #-Ptnt	Played-With Surgeon Sick-People
Hit Animal Fruit-Seed #-Agnt	Revolves Animal Fruit-Seed
Propel Fruit Fruit-Seed #-Agnt	Revolves Fruit Fruit-Seed
Propel John-Doe Car #-Agnt	Revolves John-Doe Car
Propel Orange Apple #-Agnt	Revolves Orange Apple
Propel Orange-Core Orange #-Agnt	Revolves Orange-Core Orange
Go-Down Joe-Kennedy President #-Ptnt	Sit-On Joe-Kennedy President
Eat Tom Football #-Ptnt	Sit-On Tom Football
Heavier Animal Fruit #-Agnt	Taller-Than Animal Fruit
Heavier Apple John #-Agnt	Taller-Than Apple John
Heavier Apple-Core Apple #-Agnt	Taller-Than Apple-Core Apple
Heavier Banker Money #-Agnt	Taller-Than Banker Money
Heavier Barrell Gun #-Agnt	Taller-Than Barrell Gun
Heavier Beaker Vial #-Agnt	Taller-Than Beaker Vial
Heavier Bike Man #-Agnt	Taller-Than Bike Man
Heavier Bob Football #-Agnt	Taller-Than Bob Football
Heavier Bob Rain #-Agnt	Taller-Than Bob Rain
Heavier Brain Man #-Agnt	Taller-Than Brain Man
Heavier Camp-Fire Rock #-Agnt	Taller-Than Camp-Fire Rock
Heavier Candle Paper #-Agnt	Taller-Than Candle Paper
Heavier Cannoe Man #-Agnt	Taller-Than Cannoe Man
Heavier Car Lada #-Agnt	Taller-Than Car Lada
Heavier Car Lada #-Agnt	Taller-Than Car Lada
Heavier Car Man #-Agnt	Taller-Than Car Man
Heavier Car Seat #-Agnt	Taller-Than Car Seat
Heavier Chair Wood #-Agnt	Taller-Than Chair Wood
Heavier Daddy-Bear Mommy-Bear #-Agnt	Taller-Than Daddy-Bear Mommy-Bear
Heavier Eagle Eagle-Head #-Agnt	Taller-Than Eagle Eagle-Head
Heavier Eagle-Head Eagle #-Agnt	Taller-Than Eagle-Head Eagle
Heavier Engine Car #-Agnt	Taller-Than Engine Car
Found Field Team #-Ptnt	Taller-Than Field Team
Heavier Foot Shoe #-Agnt	Taller-Than Foot Shoe
Heavier Football Bob #-Agnt	Taller-Than Football Bob
Heavier Football Field #-Agnt	Taller-Than Football Field
Heavier Football Tom #-Agnt	Taller-Than Football Tom
Found Fortress Obj_Fortress #-Ptnt	Taller-Than Fortress Obj_Fortress
Heavier Fortress Roads #-Agnt	Taller-Than Fortress Roads
Heavier Fruit Animal #-Agnt	Taller-Than Fruit Animal
Heavier Furniture Human #-Agnt	Taller-Than Furniture Human
Heavier Golf-Ball Golf-Green #-Agnt	Taller-Than Golf-Ball Golf-Green
Heavier Guitar Leadbelly #-Agnt	Taller-Than Guitar Leadbelly
Heavier Gun Barrell #-Agnt	Taller-Than Gun Barrell

Heavier Gun Tom #-Agnt	Taller-Than Gun Tom
Heavier Hammer-Handle Hammer-Head #-Agnt	Taller-Than Hammer-Handle Hammer-Head
Heavier Hammer-Head Hammer-Handle #-Agnt	Taller-Than Hammer-Head Hammer-Handle
Found Horse Human #-Ptnt	Taller-Than Horse Human
Heavier House Brick #-Agnt	Taller-Than House Brick
Heavier House Tom #-Agnt	Taller-Than House Tom
Heavier Human Clothes #-Agnt	Taller-Than Human Clothes
Heavier Human Furniture #-Agnt	Taller-Than Human Furniture
Heavier Human Horse #-Agnt	Taller-Than Human Horse
Heavier Insect Insect-Head #-Agnt	Taller-Than Insect Insect-Head
Found Insect Insect-Legs #-Ptnt	Taller-Than Insect Insect-Legs
Heavier Insect-Head Insect #-Agnt	Taller-Than Insect-Head Insect
Heavier Jfk White-House #-Agnt	Taller-Than Jfk White-House
Heavier Joe-Kennedy Jfk #-Agnt	Taller-Than Joe-Kennedy Jfk
Heavier John Apple #-Agnt	Taller-Than John Apple
Heavier John Sun #-Agnt	Taller-Than John Sun
Heavier John-Doe Car #-Agnt	Taller-Than John-Doe Car
Heavier Lada Car #-Agnt	Taller-Than Lada Car
Heavier Lada Car #-Agnt	Taller-Than Lada Car
Heavier Man Bike #-Agnt	Taller-Than Man Bike
Heavier Man Brain #-Agnt	Taller-Than Man Brain
Heavier Merlin Arthur #-Agnt	Taller-Than Merlin Arthur
Heavier Mommy-Bear Daddy-Bear #-Agnt	Taller-Than Mommy-Bear Daddy-Bear
Heavier Operating-Procedure Surgeon #-Agnt	Taller-Than Operating-Procedure Surgeon
Heavier Orange Orange-Core #-Agnt	Taller-Than Orange Orange-Core
Heavier Orange-Core Orange #-Agnt	Taller-Than Orange-Core Orange
Heavier Rain Bob #-Agnt	Taller-Than Rain Bob
Heavier Roads Fortress #-Agnt	Taller-Than Roads Fortress
Heavier Rock Camp-Fire #-Agnt	Taller-Than Rock Camp-Fire
Heavier Scissors Something #-Agnt	Taller-Than Scissors Something
Heavier Seat Car #-Agnt	Taller-Than Seat Car
Heavier Shoe Foot #-Agnt	Taller-Than Shoe Foot
Heavier Shop John #-Agnt	Taller-Than Shop John
Heavier Something Scissors #-Agnt	Taller-Than Something Scissors
Heavier Stem Flower #-Agnt	Taller-Than Stem Flower
Heavier Sun John #-Agnt	Taller-Than Sun John
Heavier Surgeon Operating-Procedure #-Agnt	Taller-Than Surgeon Operating-Procedure
Heavier Temperature-A Temperature-B #-Agnt	Taller-Than Temperature-A Temperature-B
Heavier Tom Football #-Agnt	Taller-Than Tom Football
Heavier Tom Gun #-Agnt	Taller-Than Tom Gun
Heavier Tom House #-Agnt	Taller-Than Tom House
Heavier Tom Mary #-Agnt	Taller-Than Tom Mary
Heavier Tool Human #-Agnt	Taller-Than Tool Human
Heavier Vehicle Wheel #-Agnt	Taller-Than Vehicle Wheel
Heavier Wheel Vehicle #-Agnt	Taller-Than Wheel Vehicle
Heavier White-House Jfk #-Agnt	Taller-Than White-House Jfk
Heavier Wood Chair #-Agnt	Taller-Than Wood Chair
Inhabits Army Obj_Fortress #-Ptnt	Want-Object Army Obj_Fortress
Inhabits Army Road #-Ptnt	Want-Object Army Road
Lives-In Human Object #-Ptnt	Want-Object Human Object
Inhabits Human Shoe-Sole #-Ptnt	Want-Object Human Shoe-Sole
Hit Tom Joe #-Agnt	Want-Object Tom Joe
Attack Apple Orange #-Agnt	Wear-Out Apple Orange
Attack Blade Something #-Agnt	Wear-Out Blade Something
Attack Chair Chair-Leg #-Agnt	Wear-Out Chair Chair-Leg
Attack Eagle Wings #-Agnt	Wear-Out Eagle Wings
Bounce Flesh Eagle-Torso #-Agnt	Wear-Out Flesh Eagle-Torso
Attack Gun Something #-Agnt	Wear-Out Gun Something
Attack Horse-Head Human #-Agnt	Wear-Out Horse-Head Human
Attack Horse-Torso Horse #-Agnt	Wear-Out Horse-Torso Horse
Attack Ice-Cube Temperature-A #-Agnt	Wear-Out Ice-Cube Temperature-A
Attack Orange-Peel Orange-Core #-Agnt	Wear-Out Orange-Peel Orange-Core
Propel Orchestra Listener #-Agnt	Wear-Out Orchestra Listener
Hit Roads Army #-Agnt	Wear-Out Roads Army
Attack Scisssors Scissors #-Agnt	Wear-Out Scisssors Scissors
Propel Seat Car #-Agnt	Wear-Out Seat Car
Propel Shoe Foot #-Agnt	Wear-Out Shoe Foot
Attack Shoe-Upper Foot #-Agnt	Wear-Out Shoe-Upper Foot
Attack Wall Hammer-Head #-Agnt	Wear-Out Wall Hammer-Head