REFINEMENT AND INSTITUTIONS

Marie Farrell

Supervisors: Dr Rosemary Monahan & Dr James Power

Principles of Programming Research Group

PROBLEM

❖ Difficult to combine proofs from different systems

PROPOSED SOLUTION

- ❖ Provide a theoretical framework for proof sharing
- ❖ Mathematically define each formalism
  - Including proof requirements
- ❖ Mathematically define how to integrate formalisms
- ❖ Reason about systems in the integrated formalism
  - Sharing proof components

HYPOTHESIS

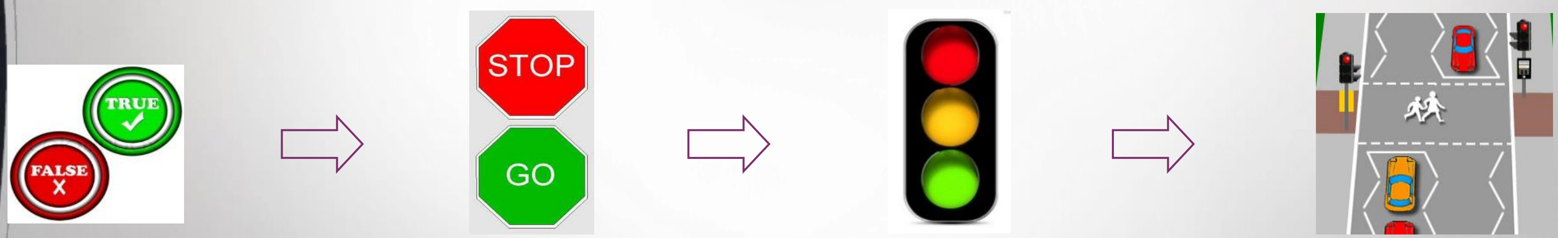
- ❖ Institutions can provide this framework
- ❖ Each formalism can be defined by an institution
- ❖ Institutions can be combined and components can be shared

Π -INSTITUTIONS

- A π-institution is a triple (Sign, ϕ, $\{Cn_\Sigma\}_{\Sigma:Sign}$) consisting of
  1. A category Sign (of signatures)
  2. A functor ϕ:Sign -> Set
  3. A consequence operator Cn_Σ
     - Σ is an object of Sign (i.e. Σ is in the alphabet)
     - Cn_Σ takes a set of axioms $A \subseteq \varphi(\Sigma)$ and gives all properties that can be deduced from A

REFINEMENT

- We model systems at different levels of abstraction



- We can map between these levels using refinement
- This process can be mathematically verified

REDUCING NONDETERMINISM

$$
\begin{aligned}
&if\ a \le b \to a \coloneqq a - b \\
&\square\ b \ge a \to b \coloneqq b - a \\
&\qquad\qquad\qquad fi
\end{aligned}
$$

$$\sqsubseteq$$

$$
\begin{aligned}
&if\ a \le b \to a \coloneqq a - b \\
&\square\ a \nleq b \to b \coloneqq b - a \\
&\qquad\qquad\qquad fi
\end{aligned}
$$

This one is nondeterministic when a=b

This one is deterministic

Classic example: Converting an NFA to a DFA

THEORIES OF REFINEMENT

- Carroll Morgan, Ralph Johan Back and Joseph Morris
- Based on Dijkstra's language of guarded commands and weakest precondition calculus.

MORGAN'S REFINEMENT

- Weakening the precondition
- Strengthening the postcondition
- Introducing local variables
- Renaming local variables
- Introducing logical constants
- Eliminating logical constants
- Expanding the frame
- Introducing `skip`
- Introducing `abort`
- Introducing assignment
- Introducing sequential composition
- Introducing alternation
- Introducing iteration

MORGAN'S REFINEMENT

Introducing alternation

$$w\colon [pre \wedge (\vee i \cdot G_i), post]$$
$$\sqsubseteq$$
$$if\,(\square i \cdot G_i \rightarrow w\colon [pre \wedge G_i, post])\,fi$$

BACK'S REFINEMENT

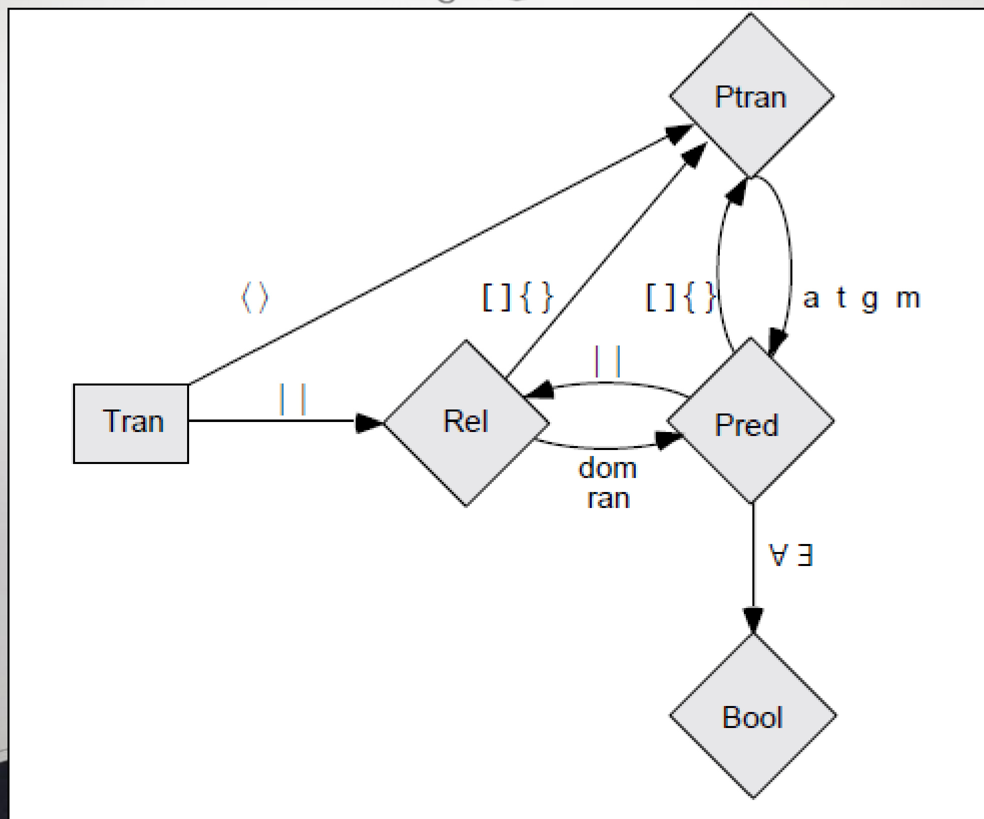


Figure: The Refinement Calculus Hierarchy

BACK'S REFINEMENT

- Similar rules to Morgan's refinement calculus
- Example
  - Introduce conditional :

$$S$$
$$\sqsubseteq$$
$$[g_1 \cup \cdots \cup g_n]; if\ g_1 \rightarrow S \,\square \ldots \square g_n \rightarrow S\ fi$$

GENERAL REFINEMENT

"The abstract entity **A** is refined by the concrete entity **C** if no user of **A** could observe if they were given **C** in its place"

Liskov Substitution

- 3 main components:
  1. Set of entities – specifications and implementations
  2. Set of contexts Ξ – the environment with which the entities interact
  3. A user – observations of a system O

$$A \sqsubseteq C$$

$$\widehat{=}$$

$$\forall x \in \Xi.\ O([C]_x) \subseteq O([A]_x)$$

LAYERS OF REFINEMENT

$$\begin{tikzcd}
\mathsf{P}_{\mathsf{H}} \arrow[r, "\sqsubseteq_{\mathsf{H}}"] & \mathsf{Q}_{\mathsf{H}} \arrow[d, "\sqsubseteq_{v}"] & \\
& \llbracket \mathsf{Q}_{\mathsf{H}} \rrbracket_{v} \arrow[r, "\sqsubseteq_{\llbracket \Xi_{\mathsf{H}}, \mathcal{O}_{\mathsf{H}} \rrbracket_{v}}"] & \mathsf{R}_{\mathsf{L}}
\end{tikzcd}$$

LAYERS OF REFINEMENT

$$
\begin{tikzcd}
\mathsf{P}_{\mathsf{H}} \arrow[rr, "\sqsubseteq_{\mathsf{H}}"] \arrow[dr, dotted, "\sqsubseteq_v"'] & & \mathsf{Q}_{\mathsf{H}} \arrow[rr, "\sqsubseteq_{\mathsf{H}}"] \arrow[dr, "\sqsubseteq_v"'] & & vA[\![\mathsf{Q}_{\mathsf{H}}]\!]_v \arrow[rr, "\sqsubseteq_{\mathsf{H}}"] & & vA(\mathsf{R}_{\mathsf{L}}) \\
& [\![\mathsf{P}_{\mathsf{H}}]\!]_v \arrow[rr, dotted, "\sqsubseteq_{[\![\Xi_{\mathsf{H}}, \mathcal{O}_{\mathsf{H}}]\!]_v}"] & & [\![\mathsf{Q}_{\mathsf{H}}]\!]_v \arrow[ur, "vA"] \arrow[rr, "\sqsubseteq_{[\![\Xi_{\mathsf{H}}, \mathcal{O}_{\mathsf{H}}]\!]_v}"] & & \mathsf{R}_{\mathsf{L}} \arrow[ur, "vA"'] &
\end{tikzcd}
$$

Refinement $\longrightarrow$ within and between layers

GALOIS CONNECTIONS

- ❖ Mathematically this vertical refinement is a Galois connection between the layers.
- ❖ Given two posets $(A, \leq_A)$ and $(B, \leq_B)$. A Galois connection between these posets consists of two maps $f: A \rightarrow B$ and $g: B \rightarrow A$, such that for all $a \in A$ and $b \in B$, we have
  - $a \leq_A f(g(a))$
  - $f(g(b)) \leq_B b$

```
MACHINE
  mac1
VARIABLES
  cars_go
  peds_go
INVARIANTS
  inv1  :  cars_go ∈ BOOL
  inv2  :  peds_go ∈ BOOL
  inv3  :  ¬(peds_go=TRUE ∧ cars_go=TRUE)
EVENTS
  INITIALISATION  ≙
     STATUS
   ordinary
  BEGIN
   act1  :  cars_go ≔ FALSE
   act2  :  peds_go ≔ FALSE
  END

  set_peds_go  ≙
     STATUS
   ordinary
  WHEN
   grd1  :  cars_go = FALSE
  THEN
   act1  :  peds_go ≔ TRUE
  END

  set_peds_stop  ≙
     STATUS
   ordinary
  BEGIN
   act1  :  peds_go ≔ FALSE
```

```
MACHINE
  mac2
REFINES
  mac1
SEES
  ctx1
VARIABLES
  cars_colour
  peds_colour
INVARIANTS
  inv1  :  peds_colour ∈ {red, green}
  inv2  :  peds_go = TRUE ⇔ peds_colour = green
  inv3  :  cars_colour ∈ {red, green}
  inv4  :  cars_go = TRUE ⇔ cars_colour = green
EVENTS
  INITIALISATION  ≙
     STATUS
   ordinary
  BEGIN
   act1  :  cars_colour ≔ red
   act3  :  peds_colour ≔ red
  END

  set_peds_green  ≙
     STATUS
   ordinary
  REFINES
   set_peds_go
  WHEN
   grd1  :  cars_colour = red
  THEN
   act1  :  peds_colour ≔ green
  END

  set_peds_red  ≙
     STATUS
   ordinary
  REFINES
   set_peds_stop
  BEGIN
   act1  :  peds_colour ≔ red
  END
```

Gluing Invariant

REFINEMENT IN JML

```java
package jmlpractice;

public class refineExamples{

    //@ public model non_null String name;
    private /*@ non_null@*/ String fullName;
    //@ private represents name = fullName;
}
```

Data Refinement

```java
public int second, minute, hour;
//@ public model long time;
//@ private represents time = second + minute*60 + hour*60*60;
```

REFINEMENT IN JML

Specification Inheritance

extends

FUTURE WORK

1. Specify a π -institution for refinement in at least two formalisms
2. Complete refinement case studies in both formalisms
3. Use π-institutions to combine proofs in these formalisms