Denotational Semantics

Part 1

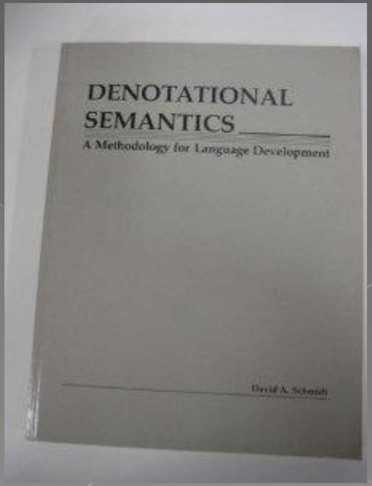
NUI MAYNOOTH
Ollscoil na hÉireann Má Nuad

IRISH RESEARCH COUNCIL
An Chomhairle um Thaighde in Éirinn

Marie Farrell IRCHSS Scholar

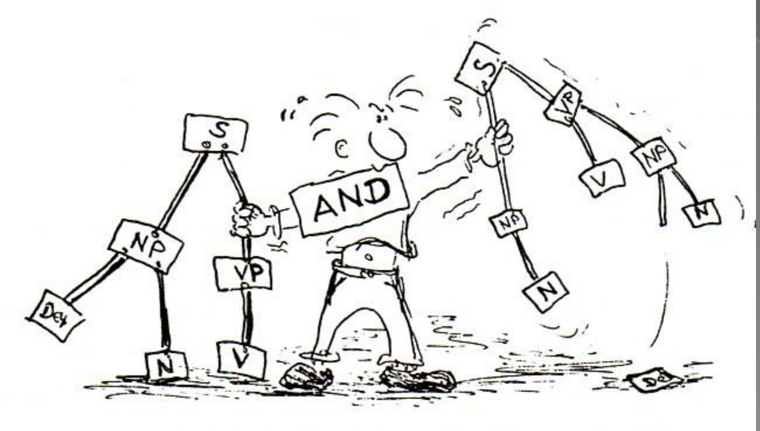
Introduction

- Syntax, semantics, pragmatics
- Mathematical objects to describe the meanings of expressions
- Only syntactically correct programs have semantics



Syntax

- Syntax definition consists of:
  - Symbols for building words
  - Word structure
  - Structure of well formed phrases
  - Sentence structure



Arithmetic – Syntax

- Symbols:
  - Digits 0-9
  - Operators + - x / ( )
- Words are numeral built from digits and operators
- Phrases are usual arithmetic expressions and the sentences are the phrases

Pascal-like programming language - Syntax

- Symbols:
  - Letters, digits, operators, brackets etc.
- Words:
  - Identifiers, numerals and operators
- Phrases:
  - Identifiers and numerals can be combined with operators to form expressions
  - Expressions can be combined with identifiers and other operators to form statements e.g. assignments, conditionals and declarations
- Sentences:
  - Statements are combined to form programs – the “sentences” of Pascal

Backus – Naur Form (BNF)

- Used to specify the internal structure of a language
- Consists of a set of equations
  - Example:
    - <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
    - <operator> ::= + | - | x | /

Nonterminals: give the name of the structural type

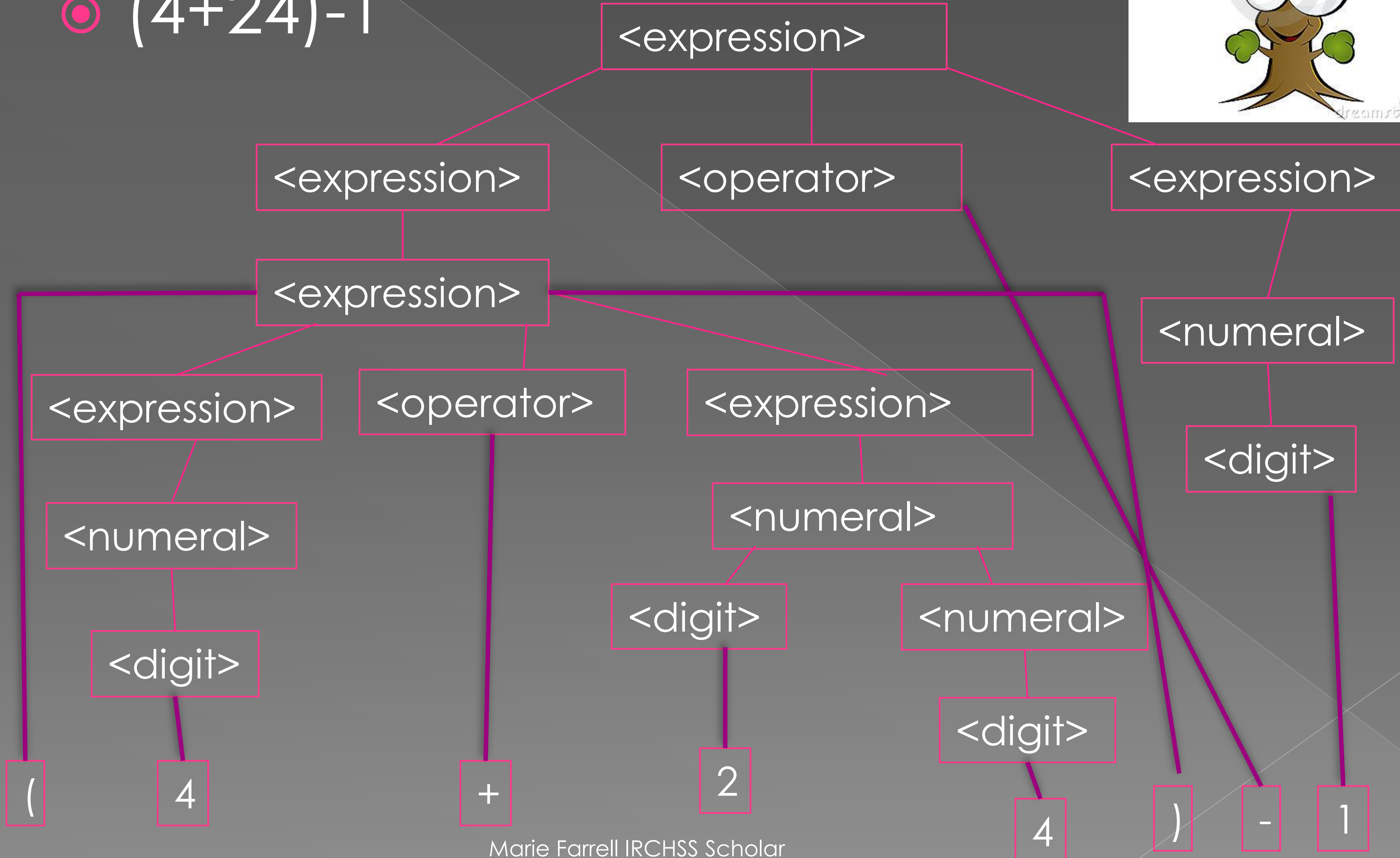
Terminal symbols: forms that belong to the structural type are built from these symbols

BNF Equations

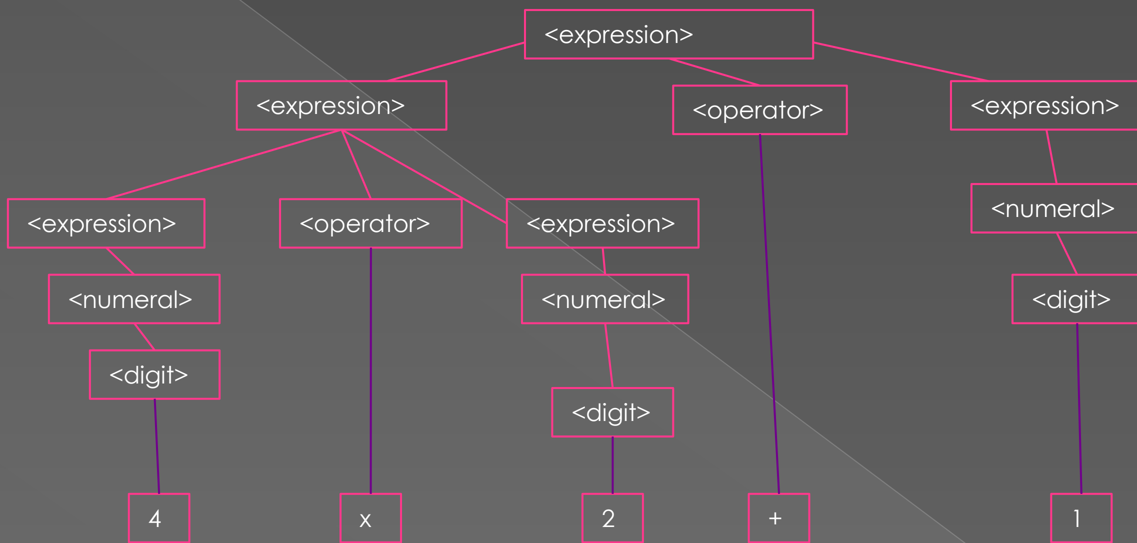
- <numeral> ::= <digit> | <digit><numeral>
- <expression> ::= <numeral> | (<expression>) | <expression> <operator><expression>

Derivation Trees

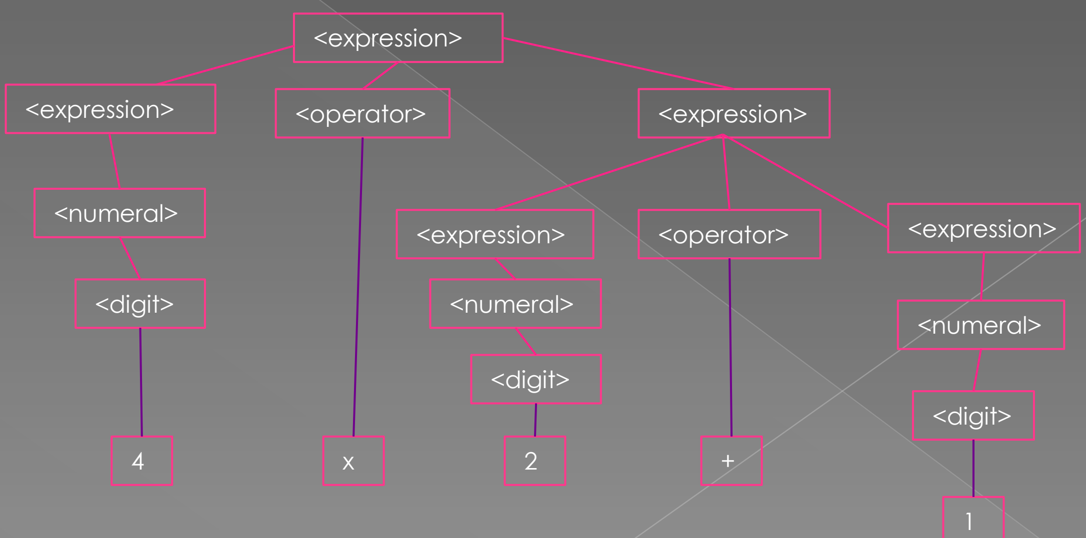
- (4+24)-1



4x2+1

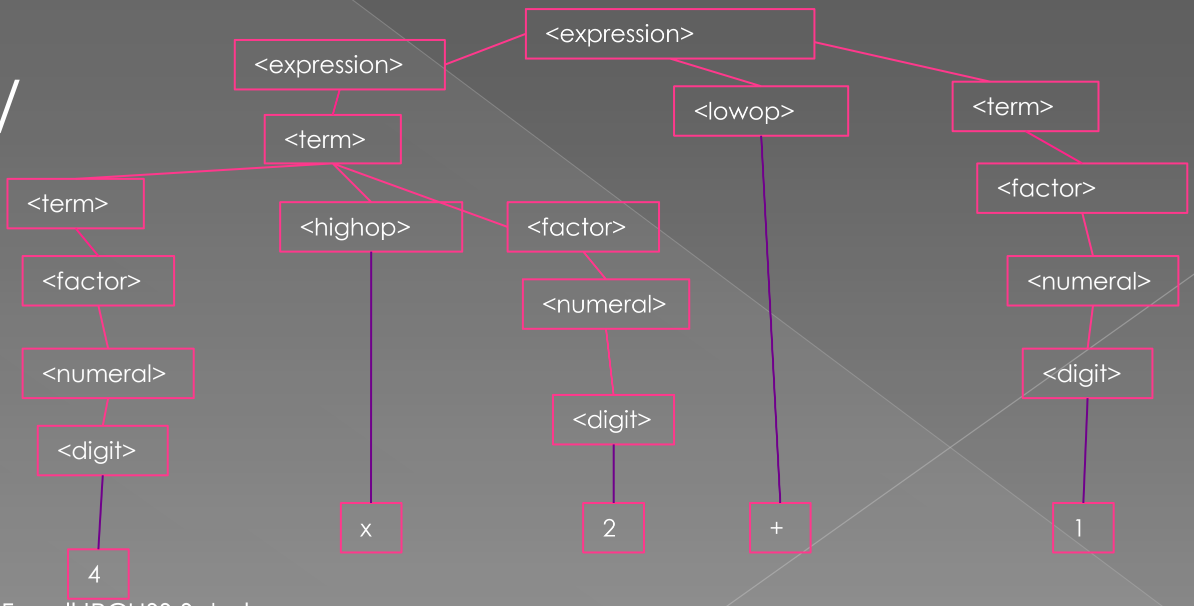


- 2 different trees
- Ambiguous



BIMDAS

- <expression> ::= <expression><lowop><term> | <term>
- <term> ::= <term><highop><factor> | <factor>
- <factor> ::= <numeral> | (<expression>)
- <lowop> ::= + | -
- <highop> ::= x | /



Abstract Syntax Definitions

- Describe structure
- Words are the building blocks, terminal symbols disappear
- Meanings are assigned to entire words, not to individual sentences

Abstract Syntax definition of Arithmetic

- <expression> ::= <numeral> | <expression><operator><expression> | left-paren<expression>right-paren
- <operator> ::= plus | minus | mult | div
- <numeral> ::= zero | one | two | ... | ninety-nine | one-hundred | ...

- The structure of arithmetic remains
- The derivation trees have the same structure as before, but the tree's leaves are the tokens instead of the symbols

Set Theory

- More abstract view of abstract syntax!
- Each nonterminal in a BNF definition names the set of those phrases that have the structure specified by the nonterminals BNF rule
- But the rule can be discarded: we introduce syntax builder operations, one for each form on the right hand side of the rule

More Arithmetic

- Sets:
  - Expression
  - Op
  - Numeral
- Operations:
  - Make-numeral-into-expression: Numeral → Expression
  - Make-compound-expression: Expression x Op x Expression → Expression
  - Make-bracketed-expression: Expression → Expression

  - Plus: Op
  - Minus: Op
  - Mult: Op
  - Div: Op

  - Zero: Numeral
  - One: Numeral
  - Two: Numeral .....

Syntax is not tied to symbols; it is a matter of structure

Syntax Domains

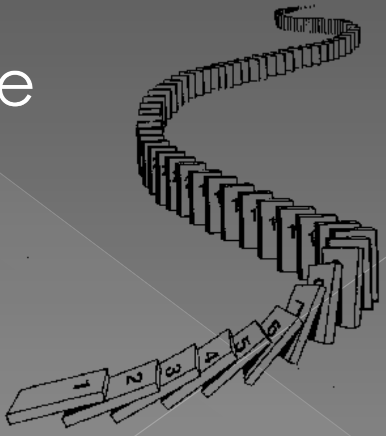
- "Syntax Domain" – a collection of values with common syntax structure.
- We specify a language's syntax by listing its syntax domains and its BNF rules
- Abstact Syntax for a File Editor:
  - P ∊ Program-session
  - S ∊ Command-sequence
  - C ∊ Command
  - R ∊ Record
  - I ∊ Identifier

  - P ::= S cr
  - S ::= C cr S | quit
  - C::= newfile | open I | moveup | moveback | insert R | delete | close

No BNF rules exist for Identifier or Record because these are collections of tokens

Mathematical and Structural Induction

- Important to be able to show that all members of a syntax domain have some common property – “structural induction”
- First, mathematical induction:
  - Show something works the first time
  - Assume it works this time
  - Show it works the next time
  - Conclusion – it works all the time.



Example

- Use induction to prove:
  - $1+4+9+\ldots+n^2 = (n(n+1)(2n+1))/6$ where $n \in \mathbb{N}$
  - Step 1: Show that n=1 holds
    - $1 = (1(1+1)(2*1+1))/6$
    - $1=1$ True
  - Step 2: Assume true for n=k
    - $1+4+9+\ldots+k^2 = (k(k+1)(2k+1))/6$
  - Step 3 : Prove true for n=k+1
    - $1+4+9+\ldots+k^2 +(k+1)^2= (k+1)(k+2)(2k+3))/6$
    - $(k(k+1)(2k+1))/6 +(k+1)^2 = (k+1)(k+2)(2k+3))/6$
    - $(2k^3+9k^2+13k+6)/6 = (2k^3+9k^2+13k+6)/6$
    - Hence, by the principle of mathematical induction the statement $1+4+9+\ldots+n^2 = (n(n+1)(2n+1))/6$ where $n \in \mathbb{N}$ is true.

Structural Induction

- The structure of mathematical induction can be formalised as a BNF rule
  - N ::= 0 | N+1
  - Any natural number is just a derivation tree
- The mathematical induction principle is a proof strategy for showing that all the trees built by the rule for N posess a property P.
- Step 1 says to show that the tree of depth 0, the leaf 0, has P.
- Step 2 says to use the fact that a tree t has property P to prove that the tree t+1 has P.
- The mathematical induction principle can be generalized to work upon any syntax domain defined by a BNF rule – “structural induction”

How does it work?

- Treating the members of a syntax domain D as trees, we show that all trees in D have property P inductively:
  - 1. Show all trees of depth 0 have P
  - 2. Assume that for an arbitrary depth $m \geq 0$ all trees of depth m or less have P
  - 3. Show that a tree of depth m+1 must have P as well.

Example

- For the domain E: Expression and its BNF rule:
    - E ::= zero | E1*E2 | (E)
  - Show that all members of Expression have the same number of left and right parentheses
  - Proof:
    - 1. zero: This is trivial.
    - 2. E1*E2: By the inductive hypothesis, E1 has m left parentheses and m right parentheses. Similarly E2 has n left parentheses and n right parentheses. Then E1*E2 has m+n left parentheses and m+n right parentheses.
    - (E): By the inductive hypothesis, E has m left parentheses and m right parentheses. Clearly, (E) has m+1 left parentheses and m+1 right parentheses.