SOM2IF15 – Compilation

How to Describe a Programming Language?

Frédéric Loulergue

2022

Outline

Levels of Description

Syntax: Which sentences are correct?

- Alphabet
- Lexical components: words or tokens
- The syntax defines the sequences of words that constitute valid sentences
- cf. Chapter “Syntactic Analysis” of this course

Levels of Description

Syntax: Which sentences are correct?

- Alphabet
- Lexical components: words or tokens
- The syntax defines the sequences of words that constitute valid sentences
- cf. Chapter “Syntactic Analysis” of this course

Semantics: What does a correct sentence mean?

- Usual description of the semantics: reference manual in English
- More precise description: formal semantics

Outline

Why the Concept of Relation Is Important?

Syntax and Semantics

- **Syntax** can be defined as a **unary relation** on a set of sequences of symbols
- Operational **semantics** will be defined as **n-ary relations**

Why the Concept of Relation Is Important?

Syntax and Semantics

- **Syntax** can be defined as a **unary relation** on a set of sequences of symbols
- Operational **semantics** will be defined as **n-ary relations**

Implementation of Programming Languages

- If an operational semantics is a deterministic relation, it can be implemented as a function
- Functional programming: interpreters by implementing operational semantics as **functions**

Why the Concept of Relation Is Important?

Syntax and Semantics

- **Syntax** can be defined as a **unary relation** on a set of sequences of symbols
- Operational **semantics** will be defined as **n-ary relations**

Implementation of Programming Languages

- If an operational semantics is a deterministic relation, it can be implemented as a function
- Functional programming: interpreters by implementing operational semantics as **functions**

Logic Programming

- Prolog programs define in essence **relations**

How to Define Relations?

By Extension/Enumeration

- Example: $A = \{ a, b, c, d \}$

\+ easy

\- not convenient if the number of tuples is large

\- impossible if it is infinite

By Comprehension

- Example: $\{ (a, f(a)) \mid a \in A \}$

\+ concise

\- something already defined is needed (here f)

\- not usable to define a relation from scratch

How to Define Relations?

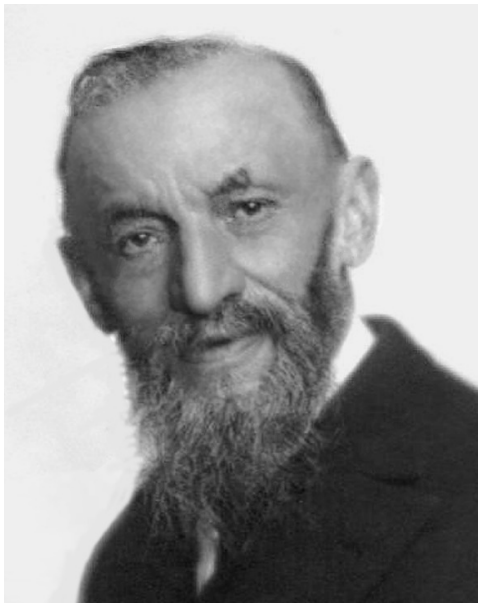
By Induction

- Base cases by enumeration
- Inductive cases: ways to build new tuples from existing ones

Natural Numbers

- O is a natural number
- if n is a natural then $S(n)$ is a natural
- Examples:

Decimal	Peano
0	O
1	$S(O)$
2	$S(S(O))$
3	$S(S(S(O))$
$\vdots$	$\vdots$



Giuseppe Peano

Inductive Definition: An Example

Addition on Peano Numbers: Function

$$O + m = m \qquad \text{for any natural number } m \qquad (1)$$

$$S(n) + m = S(n + m) \qquad \text{for any natural numbers } n,\ m \qquad (2)$$

Let's compute!

$$S(S(O)) + S(S(O)) = S(\underline{S(O) + S(S(O))}) \qquad \text{by case (2)}$$

$$= S(S(\underline{O + S(S(O))})) \qquad \text{by case (2)}$$

$$= S(S(S(S(O)))) \qquad \text{by case (1)}$$

Inductive Definition: An Example

Addition on Peano Numbers: Relation

Instead of $+$ with 2 arguments and 1 result: add as a ternary relation

- 2 first elements of the tuple correspond to the 2 arguments of $+$
- the last element corresponds to the result of the addition

Relation add

- Base case: $\mathsf{add}(O,\ m,\ m)$ for any natural number m
  To be understood as:
  the sum of O and m is m
- Inductive case: for natural numbers n, m, r,
  if $\mathsf{add}(n,\ m,\ r)$ then $\mathsf{add}(S(n),\ m,\ S(r))$
  To be understood as:
  if the sum of n and m is r then the sum of $S(n)$ and m is $S(r)$

Inductive Definition: An Example

Let's prove that 2+2=4

1. By the base case:
   $\text{add}\Big(O,\ S(S(O)),\ S(S(O))\Big)$
2. By the inductive case and 1.:
   $\text{add}\Big(S(O),\ S(S(O)),\ S(S(S(O)))\Big)$
3. By the inductive case and 2.:
   $\text{add}\Big(S(S(O)),\ S(S(O)),\ S(S(S(S(O))))\Big)$

Inference System

- Convenient way to write inductive definitions
- **Rule**: $\dfrac{j_1 \quad \dots \quad j_n}{j}$ for $n \geq 0$
- Terminology:
  - $j_1, \dots, j_n, j$: judgments
  - $j_1, \dots, j_n$: premises or hypotheses
  - j: conclusion
- If $n = 0$ we have an **axiom** (and the bar can be omitted)

Inference Systems: An Example

Natural Numbers

- $\dfrac{}{O \text{ is a natural number}}[nat_O]$
- $\dfrac{n \text{ is a natural number}}{S(n) \text{ is a natural number}}[nat_S]$

Addition as a Relation

- $\dfrac{}{\text{add}(O,\ m,\ m)}[add_O]$ with m is a natural number
- $\dfrac{\text{add}(n,\ m,\ r)}{\text{add}(S(n),\ m,\ S(r))}[add_S]$ with n, m, r are natural numbers

Inference System: Derivation Tree

Derivation Tree

A tree built using the rules of the inference system:

- the overall conclusion is the root
- the leaves are axioms

2+2=4

$$\dfrac{\dfrac{\dfrac{}{\mathsf{add}(O,\ S(S(O)),\ S(S(O)))}[add_O]}{\mathsf{add}(S(O),\ S(S(O)),\ S(S(S(O))))}[add_S]}{\mathsf{add}(S(S(O)),\ S(S(O)),\ S(S(S(S(O)))))[add_S]}$$

Outline

A Simple Imperative Language: W^0 – Syntax

Grammar

$$
\begin{array}{rcl}
\langle prog \rangle & ::= & \{\langle com \rangle ;\}^* \\
\langle com \rangle & ::= & \langle var \rangle := \langle aexp \rangle \\
& | & \texttt{if}\ \langle bexp \rangle\ \texttt{then}\ \langle prog \rangle\ \texttt{else}\ \langle prog \rangle\ \texttt{end} \\
& | & \texttt{while}\ \langle bexp \rangle\ \texttt{do}\ \langle prog \rangle\ \texttt{end} \\
\langle bexp \rangle & ::= & \texttt{true} \mid \texttt{false} \mid (\langle bexp \rangle) \mid \texttt{not}\ \langle bexp \rangle \mid \langle bexp \rangle\ \texttt{and}\ \langle bexp \rangle \\
& | & \langle aexp \rangle = \langle aexp \rangle \mid \langle aexp \rangle < \langle aexp \rangle \\
\langle aexp \rangle & ::= & \langle num \rangle \mid \langle var \rangle \mid (\langle aexp \rangle) \\
& | & \langle aexp \rangle * \langle aexp \rangle \mid \langle aexp \rangle + \langle aexp \rangle \mid \langle aexp \rangle - \langle aexp \rangle \\
\langle var \rangle & : & \text{any non empty sequence of letters and digits} \\
& & \text{starting with a letter} \\
\langle num \rangle & : & \text{any non empty sequence of digits, possibly preceded by -}
\end{array}
$$

Formalization of an Abstract Machine for W^0

Goal

- Describe how a program of W^0 can be executed, step-by-step
- We need:
  - A program, already syntactically analyzed
  - A presentation of the memory

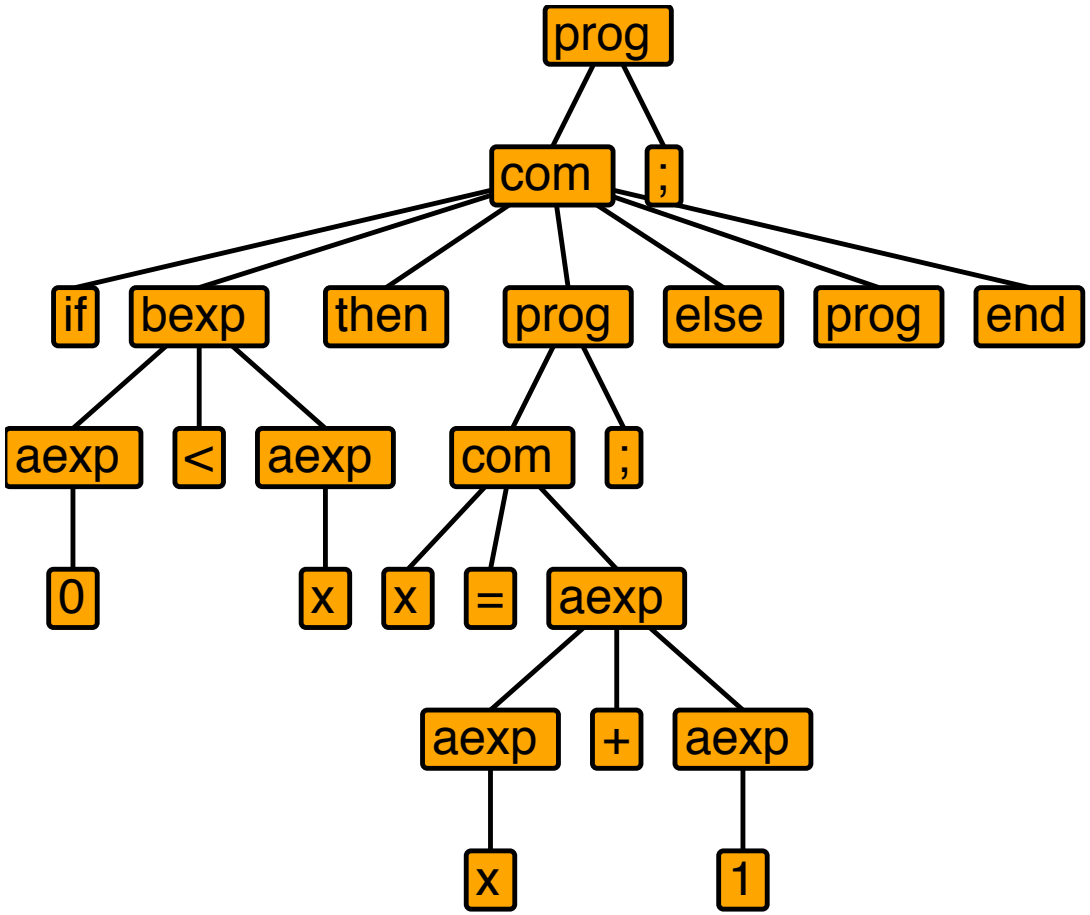
Abstract Syntax Tree

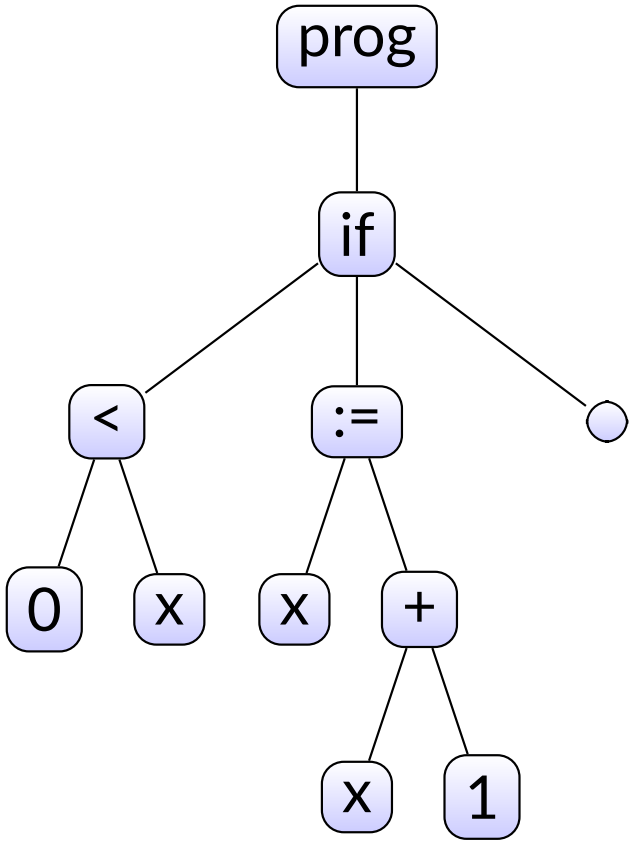
- Basically: a parse tree with all unnecessary details removed
- Not a convenient representation for writing: instead we use usual syntax + parenthesis to avoid ambiguities.

Formalization of an Abstract Machine for W^0

Parse Tree / AST

`if` $0 < x$ `then` $x := x + 1$ `else end`





Memory

Values and State

- W^0 only stores numbers in variables
- $\mathbb{Z}$: the set of numbers
- $\mathcal{X}$: the set of variables
- Memory can be represented by a **partial function**: $\sigma : \mathcal{X} \to \mathbb{Z}$
- Called **state**, **environment**, or **store**

Notation

Given σ a state, x a variable, n a number,

$$\sigma[x \mapsto n]$$

is the state that returns n if applied to x, and return the same values that σ if applied to other variables.

Memory

Example 1

If σ is the partial function such that:

- $\sigma(x) = 2$
- $\sigma(y) = 3$
- σ is undefined for all other variables

$\sigma[x \mapsto 0]$ is such that:

- $\sigma[x \mapsto 0](x) = 0$
- $\sigma[x \mapsto 0](y) = 3$
- $\sigma[x \mapsto 0]$ is undefined for all other variables

Memory

Example 2

If σ is the partial function such that:

- $\sigma(x) = 4$
- $\sigma(y) = 2$
- σ is undefined for all other variables

$\sigma[z \mapsto 1]$ is such that:

- $\sigma[z \mapsto 1](x) = 4$
- $\sigma[z \mapsto 1](y) = 2$
- $\sigma[z \mapsto 1](z) = 1$
- $\sigma[z \mapsto 1]$ is undefined for all other variables

Memory

Additional Notations

- $\sigma_\perp$ is the function undefined for all variables
  $(\sigma_\perp[x \mapsto n_1])[y \mapsto n_2]$ is only defined for x and y
- $\sigma_\perp$ can be omitted and we write
  $[x \mapsto n_1][y \mapsto n_2]$ or $[x \mapsto n_1,\ y \mapsto n_2]$

Properties

- if $x \neq y$, $(\sigma[x \mapsto n_1])[y \mapsto n_2] = (\sigma[y \mapsto n_2])[x \mapsto n_1]$
- $(\sigma[x \mapsto n_1])[x \mapsto n_2] = \sigma[x \mapsto n_2]$

Outline

Structural Operational Semantics

Goal

Give a meaning to syntax

Transition Relation

- Execution formalized as a relation between two configurations
- A configuration is a pair $\langle prog, \sigma \rangle$
- This relation, denoted by $\rightarrow$ (in infix notation), represents **one elementary step** of execution
- First we need to **evaluate** arithmetic expressions and boolean expressions

SOS: Evaluation of Numbers

Syntax / Semantics

- The *values* of arithmetic expressions are numbers in $\mathbb{Z}$
- Most basic arithmetic expressions: numbers (non-terminal $\langle num \rangle$)
- Numbers both in W^0 and usual mathematics are written as sequences of digits (0 to 9)
- To highlight the difference between syntax and semantics, let's assume for a moment that the syntax of numbers in W^0 is given by:

$$\langle num \rangle ::= \langle digit \rangle \mid \langle digit \rangle \langle num \rangle$$
$$\langle digit \rangle ::= \mathtt{0} \mid \mathtt{1}$$

- First we need a way to translate the sequence of symbols $\mathtt{101010}$ (W^0 syntax) into the number 42 belonging to $\mathbb{Z}$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\mathcal{N}[\![101010]\!]$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\mathcal{N}[\![101010]\!] = 2 \times \mathcal{N}[\![10101]\!]$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times 2 + 1)) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times 2 + 1)) + 1) \\ &= 2 \times (2 \times (2 \times 5) + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times 2 + 1)) + 1) \\ &= 2 \times (2 \times (2 \times 5) + 1) \\ &= 2 \times (2 \times 10 + 1) \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times 2 + 1)) + 1) \\ &= 2 \times (2 \times (2 \times 5) + 1) \\ &= 2 \times (2 \times 10 + 1) \\ &= 2 \times 21 \end{aligned}$$

SOS: Evaluation of Numbers

Semantic Function $\mathcal{N}$ (alternative syntax)

$$\begin{array}{lcl} \mathcal{N}[\![0]\!] & = & 0 \\ \mathcal{N}[\![1]\!] & = & 1 \end{array} \qquad \begin{array}{lcl} \mathcal{N}[\![n0]\!] & = & 2 \times \mathcal{N}[\![n]\!] \\ \mathcal{N}[\![n1]\!] & = & 2 \times \mathcal{N}[\![n]\!] + 1 \end{array}$$

Example

$$\begin{aligned} \mathcal{N}[\![101010]\!] &= 2 \times \mathcal{N}[\![10101]\!] \\ &= 2 \times (2 \times \mathcal{N}[\![1010]\!] + 1) \\ &= 2 \times (2 \times (2 \times \mathcal{N}[\![101]\!]) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times \mathcal{N}[\![10]\!] + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times (\mathcal{N}[\![1]\!]) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times (2 \times 1) + 1)) + 1) \\ &= 2 \times (2 \times (2 \times (2 \times 2 + 1)) + 1) \\ &= 2 \times (2 \times (2 \times 5) + 1) \\ &= 2 \times (2 \times 10 + 1) \\ &= 2 \times 21 \\ &= 42 \end{aligned}$$

SOS: Evaluation of Numbers

Semantics Function $\mathcal{N}$

- W^0 syntax example: 42

SOS: Evaluation of Numbers

Semantics Function $\mathcal{N}$

- W^0 syntax example: 42
- Mathematical notation: 42

SOS: Evaluation of Numbers

Semantics Function $\mathcal{N}$

- W^0 syntax example: 42
- Mathematical notation: 42
- It is possible to write $\mathcal{N}$ for such a translation

SOS: Evaluation of Numbers

Semantics Function $\mathcal{N}$

- W^0 syntax example: 42
- Mathematical notation: 42
- It is possible to write $\mathcal{N}$ for such a translation
- But it's not very interesting

SOS: Evaluation of Numbers

Semantics Function $\mathcal{N}$

- W^0 syntax example: 42
- Mathematical notation: 42
- It is possible to write $\mathcal{N}$ for such a translation
- But it's not very interesting
- Convention:
  we identify W^0 syntax and mathematical notation for *numbers*

SOS: Evaluation of Arithmetic Expressions

Semantic Function $\mathcal{A}$

- $\mathcal{A}$ takes two arguments:
  - a W^0 arithmetic expression (non-terminal $\langle aexp \rangle$)
  - an environment σ
- $\mathcal{A}$ returns a number in $\mathbb{Z}$
- In the literature, instead of writing $\mathcal{A}(e,\ \sigma)$ such an application is denoted by $\mathcal{A}[\![e]\!]\sigma$

$$\mathcal{A}[\![x]\!]\sigma = \sigma(x) \qquad \text{for any variable } x \tag{3}$$

$$\mathcal{A}[\![n]\!]\sigma = n \qquad \text{for any number } n \tag{4}$$

For any arithmetic expressions e_1 and e_2:

$$\mathcal{A}[\![e_1 * e_2]\!]\sigma = \mathcal{A}[\![e_1]\!]\sigma \times \mathcal{A}[\![e_2]\!]\sigma \tag{5}$$

$$\mathcal{A}[\![e_1 + e_2]\!]\sigma = \mathcal{A}[\![e_1]\!]\sigma + \mathcal{A}[\![e_2]\!]\sigma \tag{6}$$

$$\mathcal{A}[\![e_1 - e_2]\!]\sigma = \mathcal{A}[\![e_1]\!]\sigma - \mathcal{A}[\![e_2]\!]\sigma \tag{7}$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma \quad =$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$\mathcal{A}[\![(2 * x) + y]\!]\sigma \quad = \quad \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma \qquad \text{by (6)}$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$\begin{array}{lcll} \mathcal{A}[\![(2 * x) + y]\!]\sigma & = & \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma & \text{by (6)} \\ & = & (\mathcal{A}[\![2]\!]\sigma \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (5)} \end{array}$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$
\begin{array}{rcll}
\mathcal{A}[\![(2 * x) + y]\!]\sigma & = & \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma & \text{by (6)} \\
& = & (\mathcal{A}[\![2]\!]\sigma \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (5)} \\
& = & (2 \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (4)}
\end{array}
$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$
\begin{array}{rcll}
\mathcal{A}[\![(2 * x) + y]\!]\sigma & = & \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma & \text{by (6)} \\
& = & (\mathcal{A}[\![2]\!]\sigma \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (5)} \\
& = & (2 \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (4)} \\
& = & (2 \times 20) + \mathcal{A}[\![y]\!]\sigma & \text{by (3) and } \sigma(x) = 20
\end{array}
$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$
\begin{array}{rcll}
\mathcal{A}[\![(2 * x) + y]\!]\sigma & = & \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma & \text{by (6)} \\
& = & (\mathcal{A}[\![2]\!]\sigma \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (5)} \\
& = & (2 \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (4)} \\
& = & (2 \times 20) + \mathcal{A}[\![y]\!]\sigma & \text{by (3) and } \sigma(x) = 20 \\
& = & (2 \times 20) + 2 & \text{by (3) and } \sigma(y) = 2
\end{array}
$$

SOS: Evaluation of Arithmetic Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$
\begin{array}{rcll}
\mathcal{A}[\![(2 * x) + y]\!]\sigma & = & \mathcal{A}[\![2 * x]\!]\sigma + \mathcal{A}[\![y]\!]\sigma & \text{by (6)} \\
& = & (\mathcal{A}[\![2]\!]\sigma \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (5)} \\
& = & (2 \times \mathcal{A}[\![x]\!]\sigma) + \mathcal{A}[\![y]\!]\sigma & \text{by (4)} \\
& = & (2 \times 20) + \mathcal{A}[\![y]\!]\sigma & \text{by (3) and } \sigma(x) = 20 \\
& = & (2 \times 20) + 2 & \text{by (3) and } \sigma(y) = 2 \\
& = & 42 &
\end{array}
$$

SOS: Evaluation of Boolean Expressions

The Set $\mathbb{T}$

$\mathbb{T} = \{ T, F \}$

Operations:

- Negation: $\neg$
  $\neg T = F$
  $\neg F = T$
- Conjunction: $\wedge$
  $T \wedge T = T$
  $T \wedge F = F$
  $F \wedge T = F$
  $F \wedge F = F$

Semantic Function $\mathcal{B}$

- $\mathcal{B}$ takes two arguments:
  - a W^0 boolean expression (non-terminal $\langle bexp \rangle$)
  - an environment σ
- $\mathcal{B}$ returns a value in $\mathbb{T}$

SOS: Evaluation of Boolean Expressions

Semantic Function $\mathcal{B}$

$$\mathcal{B}[\![\texttt{true}]\!]\sigma = T \tag{8}$$

$$\mathcal{B}[\![\texttt{false}]\!]\sigma = F \tag{9}$$

$$\mathcal{B}[\![\texttt{not } b]\!]\sigma = \neg\mathcal{B}[\![b]\!]\sigma \tag{10}$$

$$\mathcal{B}[\![b_1 \texttt{ and } b_2]\!]\sigma = \mathcal{B}[\![b_1]\!]\sigma \wedge \mathcal{B}[\![b_2]\!]\sigma \tag{11}$$

$$\mathcal{B}[\![a_1 = a_2]\!]\sigma = \begin{cases} T & \text{if } \mathcal{A}[\![a_1]\!]\sigma = \mathcal{A}[\![a_2]\!]\sigma \\ F & \text{otherwise} \end{cases} \tag{12}$$

$$\mathcal{B}[\![a_1 < a_2]\!]\sigma = \begin{cases} T & \text{if } \mathcal{A}[\![a_1]\!]\sigma < \mathcal{A}[\![a_2]\!]\sigma \\ F & \text{otherwise} \end{cases} \tag{13}$$

where:

- b, b_1, b_2 are arbitrary W^0 boolean expressions
- a_1, a_2 are arbitrary W^0 arithmetic expressions

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma = 42$ by previous example

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$$\begin{array}{lcll} \mathcal{A}[\![(2 * x) + y]\!]\sigma & = & 42 & \text{by previous example} \\ \mathcal{A}[\![0]\!]\sigma & = & 0 & \text{by (4)} \end{array}$$

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma$	$=$	42	by previous example
$\mathcal{A}[\![0]\!]\sigma$	$=$	0	by (4)

therefore:

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma$	$=$	42	by previous example
$\mathcal{A}[\![0]\!]\sigma$	$=$	0	by (4)
therefore:			
$\mathcal{B}[\![(2 * x) + y < 0]\!]\sigma$	$=$	F	by (13)

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma$	$=$	42	by previous example
$\mathcal{A}[\![0]\!]\sigma$	$=$	0	by (4)

therefore:

$\mathcal{B}[\![(2 * x) + y < 0]\!]\sigma$	$=$	F	by (13)

and finally:

SOS: Evaluation of Boolean Expressions

Example

Assuming $\sigma = [x \mapsto 20,\ y \mapsto 2]$,

$\mathcal{A}[\![(2 * x) + y]\!]\sigma$	$=$	42	by previous example
$\mathcal{A}[\![0]\!]\sigma$	$=$	0	by (4)
therefore:			
$\mathcal{B}[\![(2 * x) + y < 0]\!]\sigma$	$=$	F	by (13)
and finally:			
$\mathcal{B}[\![\texttt{not}\ ((2 * x) + y < 0)]\!]\sigma$	$=$	T	by (10)

Exercise: Semantic Function $\mathcal{A}$

Evaluate

- expression $x - y$ in environment $\sigma_1 = [x \mapsto 42,\ y \mapsto 0]$
- expression $x + y$ in environment $\sigma_2 = [x \mapsto 42,\ y \mapsto 0]$
- expression $y - x$ in environment $\sigma_3 = [x \mapsto 42,\ y \mapsto 42]$

Exercise: Semantic Function $\mathcal{B}$

Evaluate

- expression $1 < n$ in environment $\sigma = [n \mapsto 3, m \mapsto 1]$
- expression $1 < n$ in environment $\sigma = [n \mapsto 2, m \mapsto 3]$
- expression $1 < n$ in environment $\sigma = [n \mapsto 1, m \mapsto 6]$

SOS: Execution of Commands/Instructions

The Relation $\rightarrow$

- Execution formalized as a binary relation between *configurations*
- Reminder: a configuration is a pair $\langle prog,\ \sigma \rangle$ where
  - *prog* is a sequence of W^0 instructions (non-terminal $\langle prog \rangle$)
  - σ is an environment: a partial function from the set of variables to $\mathbb{Z}$
- $\langle prog_1,\ \sigma_1 \rangle \rightarrow \langle prog_2,\ \sigma_2 \rangle$ means starting for a memory state σ_1, after the executing the first instruction of the list of instructions $prog_1$ the renaming instructions are $prog_2$ and the memory is in state σ_2
- We define $\rightarrow$ as an *inference system* (with axioms only)

SOS: Execution of Commands/Instructions

Transition as an Inference System

$$\frac{}{\langle x{:=}e;\, c,\ \sigma\rangle \to \langle c,\ \sigma[x \mapsto n]\rangle} \text{ where } n = \mathcal{A}[\![e]\!]\sigma \qquad \text{(assign)}$$

$$\frac{}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end};\, c,\ \sigma\rangle \to \langle c_1; c,\ \sigma\rangle} \text{ if } \mathcal{B}[\![b]\!]\sigma = T \qquad \text{(ift)}$$

$$\frac{}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end};\, c,\ \sigma\rangle \to \langle c_2; c,\ \sigma\rangle} \text{ if } \mathcal{B}[\![b]\!]\sigma = F \qquad \text{(iff)}$$

$$\frac{}{\langle \texttt{while}\ b\ \texttt{do}\ c_0\ \texttt{end};\, c,\ \sigma\rangle \to \langle \texttt{if}\ b\ \texttt{then}\ c_0; \texttt{while}\ b\ \texttt{do}\ c_0\ \texttt{end}\ \texttt{else}\ \texttt{end};\, c,\ \sigma\rangle} \qquad \text{(while)}$$

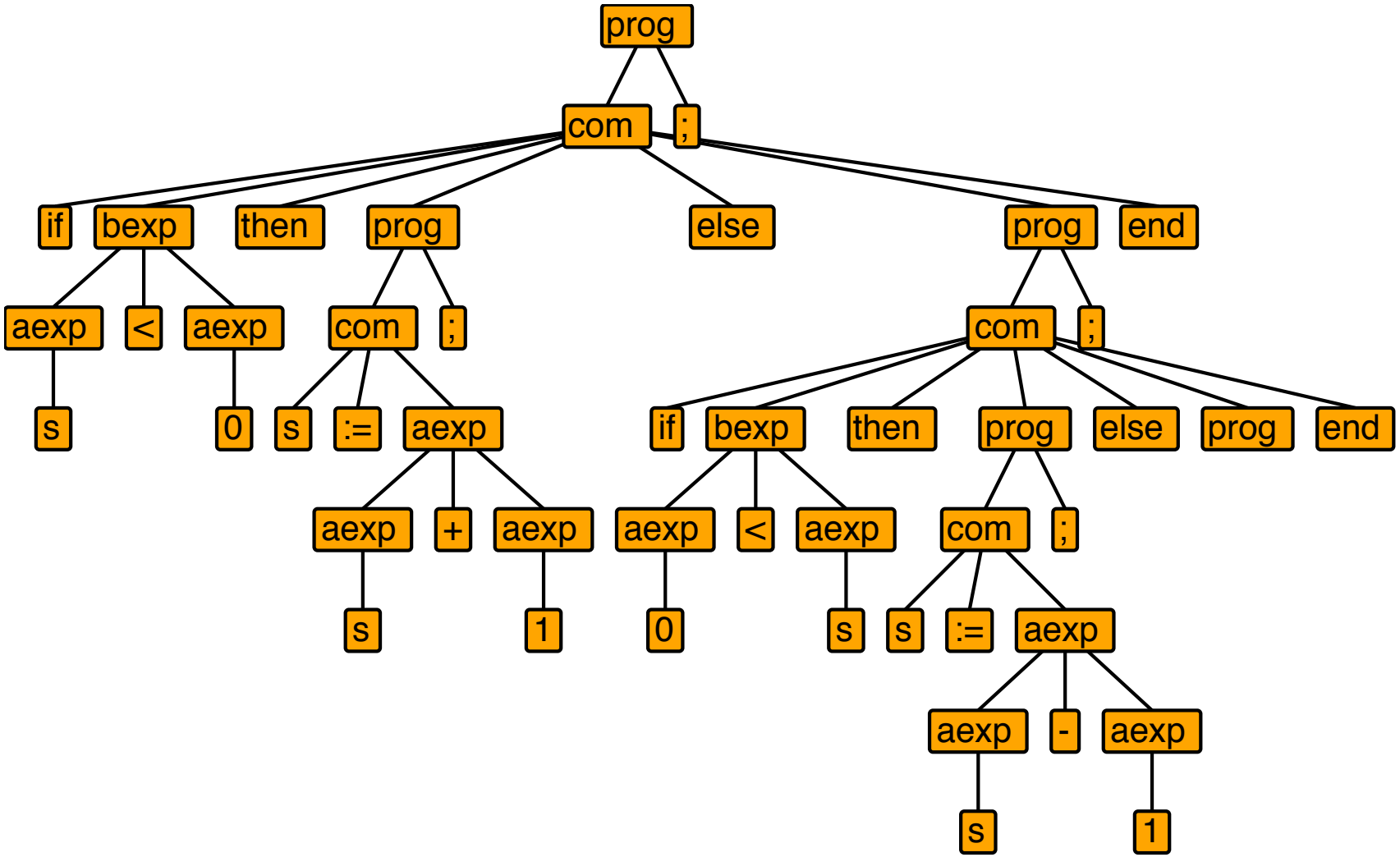
where:

- x is any W^0 variable
- e is any W^0 arithmetic expression
- b is any W^0 boolean expression
- c, c_0, c_1, c_2 are arbitrary W^0 programs (possibly empty)

SOS: Execution of Commands/Instructions

Example Program: `signal`

`if` $s < 0$ `then` $s := s + 1$; `else if` $0 < s$ `then` $s := s - 1$; `else end; end;`



SOS: Execution of Commands/Instructions

Execution Example

$\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s := s + 1; \texttt{ else if } 0 < s \texttt{ then } s := s - 1; \texttt{ else end; end;}, \sigma \rangle$

SOS: Execution of Commands/Instructions

Execution Example

$\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s := s + 1; \texttt{ else if } 0 < s \texttt{ then } s := s - 1; \texttt{ else end; end;}, \sigma \rangle$

Rule: $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end;}\, c, \sigma \rangle \rightarrow \langle c_1; c, \sigma \rangle$ (ift)

SOS: Execution of Commands/Instructions

Execution Example

$\sigma = [s \mapsto -1]$

$$\langle \texttt{if } \underbrace{s < 0}_{b} \texttt{ then } \underbrace{s := s + 1;}_{c_1} \texttt{ else } \underbrace{\texttt{if } 0 < s \texttt{ then } s := s - 1; \texttt{ else end;}}_{c_2} \texttt{ end; } \underbrace{\quad}_{c}, \sigma \rangle$$

Rule: $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}; c, \sigma \rangle \rightarrow \langle c_1; c, \sigma \rangle$ (ift)

SOS: Execution of Commands/Instructions

Execution Example

$\sigma = [s \mapsto -1]$

$\langle \texttt{if } \underbrace{s < 0}_{b} \texttt{ then } \underbrace{s{:=}s+1;}_{c_1} \texttt{ else } \underbrace{\texttt{if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end;}}_{c_2} \texttt{ end; } \underbrace{\quad}_{c}, \sigma \rangle$

Rule: $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}; c, \sigma \rangle \rightarrow \langle c_1; c, \sigma \rangle$ (ift)

$\rightarrow \langle \underbrace{s{:=}s+1;}_{c_1} \underbrace{\quad}_{c}, \sigma \rangle$ because $\mathcal{B}[\![s < 0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

SOS: Execution of Commands/Instructions

Execution Example

$\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s{:=}s+1; \texttt{ else if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end; end;}, \sigma \rangle$

$\rightarrow \langle s{:=}s+1;, \sigma \rangle$ by rule (ift) because $\mathcal{B}[\![s < 0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

SOS: Execution of Commands/Instructions

Execution Example $\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s{:=}s+1; \texttt{ else if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end; end;}, \sigma \rangle$

$\rightarrow \langle s{:=}s+1;, \sigma \rangle$ by rule (ift) because $\mathcal{B}[\![s < 0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

Rule: $\langle x{:=}e; c, \sigma \rangle \rightarrow \langle c, \sigma[x \mapsto n] \rangle$ with $n = \mathcal{A}[\![e]\!]\sigma$ (assign)

SOS: Execution of Commands/Instructions

Execution Example $\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s{:=}s+1; \texttt{ else if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end; end;}, \sigma \rangle$

$\rightarrow \langle \underbrace{s}_{x} := \underbrace{s+1}_{e}; \underbrace{\quad}_{c}, \sigma \rangle$ because $\mathcal{B}[\![s < 0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

Rule: $\langle x{:=}e; c, \sigma \rangle \rightarrow \langle c, \sigma[x \mapsto n] \rangle$ with $n = \mathcal{A}[\![e]\!]\sigma$ (assign)

SOS: Execution of Commands/Instructions

Execution Example $\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s{:=}s+1; \texttt{ else if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end; end;}, \sigma \rangle$

$\rightarrow \langle \underbrace{s}_{x} := \underbrace{s+1}_{e}; \underbrace{\quad}_{c}, \sigma \rangle$ because $\mathcal{B}[\![s<0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

Rule: $\langle x{:=}e; c, \sigma \rangle \rightarrow \langle c, \sigma[x \mapsto n] \rangle$ with $n = \mathcal{A}[\![e]\!]\sigma$ (assign)

$\rightarrow \langle \underbrace{\quad}_{c}, \sigma[s \mapsto 0] \rangle$ because $\mathcal{A}[\![s+1]\!]\sigma = \sigma(s) + 1 = 0$

SOS: Execution of Commands/Instructions

Execution Example $\sigma = [s \mapsto -1]$

$\langle \texttt{if } s < 0 \texttt{ then } s{:=}s+1; \texttt{ else if } 0 < s \texttt{ then } s{:=}s-1; \texttt{ else end; end;}, \sigma \rangle$

$\rightarrow \langle s{:=}s+1;, \sigma \rangle$ by rule (ift) because $\mathcal{B}[\![s < 0]\!]\sigma = T$ because $\mathcal{A}[\![s]\!]\sigma = -1$

$\rightarrow \langle \epsilon, \sigma[s \mapsto 0] \rangle$ by rule (assign) because $\mathcal{A}[\![s+1]\!]\sigma = \sigma(s) + 1 = 0$

where ϵ is used to represent the empty sequence of instructions

Exercise: Transition Relation $\rightarrow$

SOS

- Execute the following program: $x{:=}x - y; y{:=}x + y; x{:=}y - x;$
- starting from a state $\sigma = [x \mapsto 42, y \mapsto 0]$

Exercise: Transition Relation $\rightarrow$

SOS

- Execute the following program:
  $m{:=}1; \texttt{while}\ 1 < n\ \texttt{do}\ m{:=}m * n; n{:=}n - 1;\ \texttt{end};$
- starting from a state $\sigma = [n \mapsto 3]$

Outline

Natural Semantics

Difference between SOS and Natural Semantics

- SOS formalizes execution step-by-step
- Natural semantics is a relation between an initial configuration and a **final** state
- The judgments for this kind of semantics have the form:

$$\langle c,\ \sigma \rangle \rightarrow \sigma'$$

A Natural Semantics for W^0

$$\frac{}{\langle \epsilon,\ \sigma\rangle \to \sigma} \quad \text{(noop)}$$

$$\frac{}{\langle x{:=}e;\ ,\ \sigma\rangle \to \sigma[x \mapsto n]} \text{ where } n = \mathcal{A}[\![e]\!]\sigma \quad \text{(assign)}$$

$$\frac{\langle c_1,\ \sigma\rangle \to \sigma' \qquad \langle c_2,\ \sigma\rangle' \to \sigma''}{\langle c_1\ c_2,\ \sigma\rangle \to \sigma''} \quad \text{(sequence)}$$

$$\frac{\langle c_1,\ \sigma\rangle \to \sigma_1}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end};\ c,\ \sigma\rangle \to \sigma_1} \text{ if } \mathcal{B}[\![b]\!]\sigma = T \quad (\text{if}_t)$$

$$\frac{\langle c_2,\ \sigma\rangle \to \sigma_2}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end};\ c,\ \sigma\rangle \to \sigma_2} \text{ if } \mathcal{B}[\![b]\!]\sigma = F \quad (\text{if}_f)$$

$$\frac{}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end};\ ,\ \sigma\rangle \to \sigma} \text{ if } \mathcal{B}[\![b]\!]\sigma = F \quad (\text{while}_f)$$

$$\frac{\langle c,\ \sigma\rangle \to \sigma' \qquad \langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end};\ ,\ \sigma'\rangle \to \sigma''}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end};\ ,\ \sigma\rangle \to \sigma''} \text{ if } \mathcal{B}[\![b]\!]\sigma = T \quad (\text{while}_t)$$

where: x is any variable, e, b are any arithmetic/boolean expressions, c, c_1, c_2 are arbitrary W^0 programs

Outline

The W^1 Language

Grammar

$$
\begin{array}{rcl}
\langle prog \rangle & ::= & \{\langle decl \rangle ;\}^* \ \{\langle com \rangle ;\}^* \\
\langle decl \rangle & ::= & \texttt{var}\ \langle identifier \rangle\ \texttt{is}\ \langle type \rangle \\
\langle type \rangle & ::= & \texttt{integer} \mid \texttt{boolean} \\
\langle com \rangle & ::= & \langle identifier \rangle := \langle exp \rangle \\
 & \mid & \texttt{if}\ \langle exp \rangle\ \texttt{then}\ \{\langle com \rangle ;\}^*\ \texttt{else}\ \{\langle com \rangle ;\}^*\ \texttt{end} \\
 & \mid & \texttt{while}\ \langle exp \rangle\ \texttt{do}\ \{\langle com \rangle ;\}^*\ \texttt{end} \\
\langle exp \rangle & ::= & \texttt{true} \mid \texttt{false} \mid (\langle exp \rangle) \mid \texttt{not}\ \langle exp \rangle \mid \langle exp \rangle\ \texttt{and}\ \langle exp \rangle \\
 & \mid & \langle exp \rangle = \langle exp \rangle \mid \langle exp \rangle < \langle exp \rangle \mid \langle num \rangle \mid \langle identifier \rangle \\
 & \mid & \langle exp \rangle * \langle exp \rangle \mid \langle exp \rangle + \langle exp \rangle \mid \langle exp \rangle - \langle exp \rangle \\
\langle identifier \rangle & : & \text{any non empty sequence of letters and digits} \\
 & & \text{starting with a letter} \\
\langle num \rangle & : & \text{any non empty sequence of digits,} \\
 & & \text{possibly preceded by “-”}
\end{array}
$$

Typing

The Ingredients

- For an expression: we need its type (`integer` or `boolean`)
- For a command: we need to know it's well typed, but it doesn't have a type
- For a program: same as for a command
- We will use a type environment Γ that maps identifiers to their types as *declared*
- Same notations for environments and type environments

Typing

Judgements for Typing Expressions

$$\Gamma \vdash e : \tau$$

reads "in type environment Γ the expression e has type τ"

Judgements for Typing Commands and Programs

- $\Gamma \vdash_c com$: the command com is well typed in typing environment Γ
- $\vdash_p c$: the program c is well typed

Typing Expression

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

$$\frac{}{\Gamma \vdash n : \texttt{integer}}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{boolean}} \qquad \frac{}{\Gamma \vdash \texttt{false} : \texttt{boolean}}$$

$$\frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 = e_2 \ : \ \texttt{boolean}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{integer} \qquad \Gamma \vdash e_2 : \texttt{integer}}{\Gamma \vdash e_1 \oplus e_2 \ : \ \texttt{integer}} \text{ with } \oplus \in \{+, -, *\}$$

$$\frac{\Gamma \vdash e_1 : \texttt{boolean} \qquad \Gamma \vdash e_2 : \texttt{boolean}}{\Gamma \vdash e_1 \ \texttt{and} \ e_2 \ : \ \texttt{boolean}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{integer} \qquad \Gamma \vdash e_2 : \texttt{integer}}{\Gamma \vdash e_1 < e_2 \ : \ \texttt{boolean}}$$

Typing Commands and Programs

Commands

$$\frac{\Gamma(x) = \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash_c x{:=}e}$$

$$\frac{\Gamma \vdash e : \texttt{boolean} \qquad \forall com \in c_1, \Gamma \vdash_c com \qquad \forall com \in c_2, \Gamma \vdash_c com}{\Gamma \vdash_c \texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}}$$

$$\frac{\Gamma \vdash e : \texttt{boolean} \qquad \forall com \in c, \Gamma \vdash_c com}{\Gamma \vdash_c \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}}$$

Programs

$$\frac{\Gamma = \text{make}(decls) \qquad \forall com \in c, \Gamma \vdash_c com}{\vdash_p decls\ c}$$

where **make** creates the environment Γ from declarations