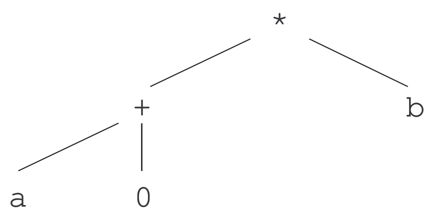
CS611 Lecture 1: IMP 8/27/99
Maasha

Concrete vs. Abstract syntax

Lisp sexpression	C expression
$(*, (+, a, 0), b)$	$((a + 0) * b)$

The above statements in different languages both have the same abstract syntactic structure represented by the tree below:



The course will use abstract syntax as it circumvents dealing with the particulars of parsing. Expressions used in the course will be look similar to instances of concrete syntax but instead will refer to abstract syntax instances (e.g. parenthesis's do not occur in the abstract syntax but may be inserted parenthesis as needed to disambiguate parsing).

IMP

Consider the simple but Turing complete language IMP[1]. In IMP there are the following syntactic groupings:

a: arithmetic expressions
n: whole number
X: location - variables

Theese groupings have the following productions:

$a := n|X|a_0 + a_1|a_0 \times a_1|a_0 - a_1$
$b := a_0 = a_1|a_0 \leq a_1|b_0 \wedge b_1|b_0 \vee b_1|\neg b$
$c := skip|X := a|c_1; c_2|if\, b\, c_1\, else\, c_2|while\, b\, do\, c$

The above formation rules describe how to form valid IMP expression trees.

State functions

A *state function* is a mapping from locations to numbers:
$\sigma : X \rightarrow N$
where σ applied to location X yields and integer n expressed by the following evaluation relation:
$\sigma(X) \rightarrow n$

Axioms

[1] Winskel A.1

In the case of the state function the evaluation rule does not have an assumption; the class of rules without premises are termed *axioms*. The following rule is an axiom:

$$\frac{}{\langle n,\, \sigma \rangle \rightarrow n}$$

Inference Rules

Not all rules are without assumptions; rules for evaluations with assumptions are given by *inference rules*. The notation of inference rules places the *conclusion* below the line and the facts or *premises* used to derive the conclusion below the line. To the right of the inference are sometimes equations that constrain the application of a rule. The constraint equations unlike premises need not be derived by a proof. Inference rules for a few arithmetic expressions are given below:

$$\frac{\sigma(\mathrm{X}) \rightarrow \mathrm{n}}{\langle n,\, \sigma \rangle \rightarrow n}$$

$$\frac{\langle a_0,\, \sigma \rangle \rightarrow n_1 \qquad \langle a_1,\, \sigma \rangle \rightarrow n_2}{\langle a_0 + a_1,\, \sigma \rangle \rightarrow n_3} \qquad n_1 + n_2 = n_3$$

$$\frac{\langle a_0,\, \sigma \rangle \rightarrow n_1 \qquad \langle a_1,\, \sigma \rangle \rightarrow n_2}{\langle a_0 \times a_1,\, \sigma \rangle \rightarrow n_3} \qquad n_1 \times n_2 = n_3$$

$$\frac{\langle a_0,\, \sigma \rangle \rightarrow n_1 \qquad \langle a_1,\, \sigma \rangle \rightarrow n_2}{\langle a_0 - a_1,\, \sigma \rangle \rightarrow n_3} \qquad n_1 - n_2 = n_3$$

Metavariables

The rules for evaluation mention *metavariables* (e.g. n, X, a) that are instantiated to give a rule instance. A rule is applied by finding bindings for all metavariables such that the premises are satisfied. The rule then allows the conculsion to be derived.

<u>Example</u>

Consider evaluating the expression:
$\langle (d + 2) \times 3,\, \sigma \rangle \rightarrow ?$
with the state function: $\sigma_1 = \{(d, 4)\}$
the following evaluation can be derived:

$$\frac{\dfrac{\langle d,\, \sigma \rangle \rightarrow 4 \qquad \langle 2,\, \sigma \rangle \rightarrow 2}{\langle (d + 2),\, \sigma \rangle \rightarrow 6} \qquad \langle 3,\, \sigma \rangle \rightarrow 3}{\langle (d + 2) \times 3,\, \sigma \rangle \rightarrow 18}$$

Commands

The evaluation of a command yields a new state:
$\langle c,\, \sigma \rangle \rightarrow \sigma'$

The inference rules for assignment:

$$\frac{\langle a,\, \sigma \rangle \rightarrow n}{\langle X := a,\, \sigma \rangle \rightarrow \sigma'}$$

Where σ' is given by:

$\sigma'(Y) = \{ {}^{Y \neq X : \sigma(Y)}_{Y = X : n}$

Which is given by the following notations: $\sigma' = \sigma[X \backslash n]$ or $\sigma' = \sigma[x \mapsto n]$

Sequencing for commands is given by the following inference rule:

$$\frac{\langle c_0,\, \sigma\rangle \rightarrow \sigma' \qquad \langle c_1,\, \sigma'\rangle \rightarrow \sigma''}{\langle c_0; c_1,\, \sigma\rangle \rightarrow \sigma''}$$

$\square$