Notes on Type Checking - 1

Version 2

SERC/IIITH

March, 2015

The main reference for the present material is [1]. Students must note that test questions will depend not only on these skeletal notes but also the material covered in class.

1 The Language $L(num)$

We will consider set the Exp of expressions e as described below.

	Abstract Syntax	Concrete Syntax	Description
$e ::=$			
	$num[n]$	n	numeral
	$plus(e_1; e_2)$	$e_1 + e_2$	addition

1.1 The size of an expression

We define the size function $|.| : Exp \to \mathbb{N}$ by induction on Exp.

1. $|num[n]| = 1$
2. $|plus(e_1; e_2)| = 1 + |e_1| + |e_2|$

For example, $|plus(num[7]; e)| = 2 + |e|$.

1.2 Problem

Q: What is the meaning of these expressions?

A: The meaning is described in terms of the transitions of an abstract *transition system* as described next.

1.3 Transition Systems

A transition system TS consists of

1. A set of states S.
2. A subset of initial states $S_{init} \subseteq S$.
3. A subset of final states $S_{fin} \subseteq S$.
4. A binary relation $\mapsto \subseteq S \times S$ on states.

If $a \mapsto b$ we call this a transition, or reduction, or assertion, or judgment. The relation $\mapsto$ is sometimes given completely at one go as a subset of $S \times S$. However, frequently the relation $\mapsto$ needs to be generated by giving some initial axioms and derivation rules. We will see an example soon.

A TS is said to be deterministic if each state either can not be reduced or reduces to a unique state.

Let $\mapsto^*$ represent the iteration of the binary relation $\mapsto$. This extended relation $\mapsto^*$ is defined by the three following judgment derivation rules.

1. $\frac{}{s \mapsto^* s}$
2. $\frac{s \mapsto^* s' \quad s' \mapsto s''}{s \mapsto^* s''}$
3. $\frac{s \mapsto s' \quad s' \mapsto^* s''}{s \mapsto^* s''}$

Question: Are both the properties 2 and 3 above necessary to be stated? Would it be possible to deduce either one from the other?(Hint: Define the size of $a \mapsto^* b$ and use induction.)

1.4 Structural Dynamics

The structural dynamics for the language $L(num)$ is given by the transition systems whose states are expressions. Any expression can be an initial one, and the final states are *values*, which represent completed computations. The judgments are given by the derivation rules listed below. The first one is an axiom.

1.4.1 Value Judgments

$$\frac{}{num[n] \quad val}$$

1.4.2 Transition Judgments

The first one gives the primitive application $PLUS_N$ or P_N.

$$\frac{n_1 + n_2 = n_3 \qquad nat}{plus(num[n_1]; num[n_2]) \mapsto num[n_3]} \quad P_N$$

This rule can also be stated as an axiom.

$$\frac{}{plus(num[n_1]; num[n_2]) \mapsto num[n_1 + n_2]} \quad P_N$$

The next two judgment derivation rules are concerned with the order of evaluation. L stands for 'left' and R stands for 'right'.

$$\frac{e_1 \mapsto e_1'}{plus(e_1; e_2) \mapsto plus(e_1'; e_2)} \quad P_L$$

$$\frac{e_1 \quad val \qquad e_2 \mapsto e_2'}{plus(e_1; e_2) \mapsto plus(e_1'; e_2')} \quad P_R$$

Question: Why is it necessary to add the condition that e_1 is a value in the judgment derivation rule P_R when such a condition was not stated for P_L?

1.5 Semantics

The semantics of an expression e_0 is defined by a transition sequence

$$e_0 \mapsto e_1 \mapsto e_2 \mapsto \cdots .$$

Each step $e_i \mapsto e_{i+1}$ is derived using one of the judgment derivation rules listed above.

Example:

$\underline{Expression}$	$\underline{Remarks}$
$plus(plus(num[2]; num[3]); plus(num[6]; num[7]))$	$initial$
$\mapsto plus(num[5]; plus(num[6]; num[7]))$	$P_N \& P_L$
$\mapsto plus(num[5]; num[13])$	$P_N \& P_R$
$\mapsto num[18]$	P_N

We can rewrite the above example using concrete syntax.

$\underline{Expression}$	$\underline{Remarks}$
$(2+3)+(6+7)$	$initial$
$\mapsto 5+(6+7)$	$P_N\ \&P_L$
$\mapsto 5+13$	$P_N\&P_R$
$\mapsto 18$	P_N

Question: Can we reduce the expression by starting with $(2+3)+13$?

References

[1] Harper, R.: *Practical Foundations for Programming Languages*, Cambridge University Press, December 2012.