Notes on Type Checking - 2

Version 1

SERC/IIITH

March, 2015

Students must note that test questions will depend not only on these skeletal notes but also the material covered in class.

1 The Language $L(num)$

The main reference for this section is Chapter 5 of the book by Harper [2].

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.

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

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 finite or infinite 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.

A transition sequence as above is also called a run.

An expression e is said to be irreducible or normal if there is no e' such that $e \mapsto e'$. Otherwise e is said to be reducible. According to our definition of $L(num)$, the values $num[n]$ are irreducible.

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 $(2+3)+13$?

To simplify the writing from now on we replace $plus(num[n_1]; num[n_2])$ by $p(n[n_1]; n[n_2])$.

1.6 Three Properties of $L(num)$

We can ask three questions for any transition system.

Termination Is every run finite?

Finality What are all the irreducible elements?

Determinacy If $e \mapsto e'$ and $e \mapsto e''$, then is $e' = e''$?

We now answer these questions for the transition system associated to $L(num)$ and prove three properties. We show that the system is terminating and deterministic and that the irreducible elements are precisely the values.

1.7 Termination

Recall the definition of size of an expression $|e|$ from Section 1.1.

Termination is proved by proving the theorem below. In the language of Dijkstra [1] the size is a bound function. It is also called a measure function in some texts.

Theorem 1.1 *If* $e \mapsto e'$*, then* $|e'| < |e|$.

Proof The proof is by rule induction. In our case since the rules define the transition judgments we could say that we perform induction on $\mapsto$.

The judgment $e \mapsto e'$ can arise in three ways. In each of these cases we show that if the claim holds for the premises then it holds for the conclusion.

Case 1 Suppose $\frac{}{e \mapsto e'} \quad P_N$.

By pattern matching with the P_N rule we see that e must be of the form $p(n[n_1]; n[n_2])$ and e' must be of the form $n[n_1 + n_2]$. But then $|e| = 3$ and $|e'| = 1$. Since $|e'| < |e|$ we are done.

Case 2 Suppose $\frac{}{e \mapsto e'} \quad P_L$.

By pattern matching we conclude that $e = p(e_1'; e_2)$, $e_1 \mapsto e_1'$, and $e' = p(e_1'; e_2)$. By Induction Hypothesis $|e_1'| < |e_1|$. We then have

$$|e'| = |p(e_1'; e_2| = 1 + |e_1'| + |e_2| < 1 + |e_1| + |e_2| = |p(e_1; e_2) = |e|.$$

Case 3 Suppose $\frac{}{e \mapsto e'} \quad P_R$.

This case is left to the reader as an exercise. $\square$

1.8 Finality

Theorem 1.2 *An expression e in $L(num)$ is irreducible if and only if it is a value.*

Proof The proof is by inspection of the rules.

If e is a value then e must be irreducible because there is no rule that gives a judgment $a \mapsto b$ in which a is a value.

Next, we must prove that if e is irreducible then e is a value. Contrapositively, we should prove that if e is not a value then e is reducible. Let then e be not a value. Then e must be of the form $p(e_1; e_2)$. The proof is by induction on expressions.

Case 1.1 e_1 is a value and e_2 is a value.
Then $e = p(e_1; e_2) = p(n[n_1]; n[n_2]) \mapsto n[n_1 + n_2]$, so that e is reducible.

Case 1.2 e_1 is a value and e_2 is not a value.
By the Induction Hypothesis e_2 is reducible. Say, $e_2 \mapsto e_2'$. Recalling the rule P_R we see that

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

Hence $e = p(e_1; e_2)$ is reducible.

Case 2 e_1 is not a value. By the Induction Hypothesis e_1 is reducible. Let $e_1 \mapsto e_1'$. Recalling the rule P_L we see that

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

Hence $e = p(e_1; e_2)$ is reducible.

The theorem is proved. □

We see from the above that to prove the three properties for even the very simple language $L(num)$ we need to consider several cases. It is then evident that in the case of standard languages there will always be very many cases to consider. This suggests the need of a theorem prover.

1.9 Determinacy

Theorem 1.3 *If* $e \mapsto e'$ *and* $e \mapsto e''$*, then* $e' = e''$*.*

Proof The proof is by induction on the derivations of the two transitions above. Since each transition can arise in three ways by each of the rules P_N, P_L and P_R we have nine cases to consider. We give the proof here in three cases leaving the remaining six to the reader.

Suppose the transition $e \mapsto e'$ is derived from the rule P_N. Then $e = p(n[n_1]; n[n_2]) \mapsto n[n_1 + n_2]$, so that $e' = n[n_1 + n_2]$.

Case 1 Let the transition $e \mapsto e''$ be derived from the rule P_N. Then it is clear that we must have $e = p(n[n_1]; n[n_2]) \mapsto n[n_1 + n_2]$, so that $e'' = n[n_1 + n_2]$. Hence $e' = e''$.

Case 2 Let the transition $e \mapsto e'$ be derived from the rule P_L. Then e must be $p(e_1; e_2)$ so that $e_1 = n[n_1]$, $e_2 = n[n_2]$. But we must also have $e_1 \mapsto e_1'$, implying that e_1 is reducible which is not the case. So the P_L rule can not apply.

Case 3 An argument similar to the above would imply that $e_2 = n[n_2]$ is reducible which is not the case. So this case also does not apply.

The theorem is proved. □

Questions:

1. Consider a language $L(num, plus, minus, mult, div)$ of numbers with all four arithmetic operations. How would you write down the derivation rules? How would you answer the three questions of termination,finality, and determinacy?
2. Answer the same three question for the language of numbers and strings considered by Harper.

References

[1] Dijkstra, E.W.: *A Discipline of Programming*, Prentice Hall of India, 1979.

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