Lecture on Transition Systems [Revision Hour]

CS 4271
Abhik Roychoudhury
National University of Singapore

Caution

- The lecture slides may appear to be a different format.
- This is because they are taken from my earlier set of slides in LaTeX-PDF

Material

- Details of state evolution of sequential programs for a toy Programming Language
  - This combined with the modeling of concurrency discussed today allows for modeling of non-trivial concurrent systems using Kripke Structures.
  - This material is not for examination purposes, it is meant to enhance our understanding of the state machine underlying to a sequential program.

Operational Semantics

- Operational Semantics clarifies the execution model of a program.
- Closes the gap between the text of a program and the behaviors represented by it.
- Let us look only at sequential programs for the moment.

IMP : a toy imperative language

- **IMP** is an imperative language in the style of PASCAL or C (even though some of the syntax may be different)
- The language contains arithmetic and boolean expressions as well as if-then-else, while statements.
- The syntax of the program will be described by BNF grammars.

IMP : a toy imperative language

- During execution of IMP program, the state of execution will be captured by the values of program variables.
- Operational semantics will be described by rules which specify how
  - Expressions in IMP pgm. are evaluated
  - Statements in IMP pgm. change the state

Syntax of IMP

- non-negative integers N
- truth values $T = \{\text{true, false}\}$
- variables V
- arithmetic expressions A
- boolean expressions B
- statements/commands C

Syntax of Arithmetic and Boolean expressions

$$A \rightarrow N$$
$$A \rightarrow V$$
$$A \rightarrow A + A$$
$$A \rightarrow A * A$$

$$B \rightarrow T$$
$$B \rightarrow A = A$$
$$B \rightarrow A \leq A$$
$$B \rightarrow \neg B$$
$$B \rightarrow B \wedge B$$

Syntax of Commands C

$$C \rightarrow skip$$
$$C \rightarrow V := A$$
$$C \rightarrow C; C$$
$$C \rightarrow if\ B\ then\ C\ else\ C$$
$$C \rightarrow while\ B\ do\ C$$

Execution model

- Operational semantics of IMP describes how programs in that language are excuted.
- To describe this, it needs to assume an underlying execution model.
- The execution model could be thought as a state machine although not necessarily a finite state machine.

Operational Semantics

Operational Semantics for the IMP language will give rules to describe the following:

Given a state s

- how to evaluate arithmetic expressions
- how to evaluate boolean expressions
- how the commands can alter s to a new state s'

States

A state is a valuation of program variables i.e. each variable is mapped to a value in its type.

- Thus, if $\{a, b\}$ are the only variables in an IMP program, then each of the following are states in the execution model
  - a = 0, b = 0
  - a = 0, b = 1
  - a = 0, b = 2
  - ...
  - a = 1, b = 0
  - ...

Meaning of Arith. Expressions A - (1)

- Numbers: $\langle n, s\rangle \equiv n$
  Number n in any state s evaluates to n
  e.g. $\langle 0, s\rangle \equiv 0$, $\langle 1, s\rangle \equiv 1$
- Variables: $\langle X, s\rangle \equiv s(X)$
  Variable X in state s evaluates to value of X in s.
  e.g. $\langle a, (a = 5, b = 20)\rangle \equiv 5$
  $\langle b, (a = 5, b = 20)\rangle \equiv 20$

Meaning of Arith. Expressions A - (2)

- Sums:

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 + a_1, s\rangle \equiv n} \text{ where } n \text{ is the sum of } n_0 \text{ and } n_1$$

e.g. $\langle a + b,\ (a = 5, b = 20)\rangle \equiv 25$

- Products:

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 * a_1, s\rangle \equiv n} \text{ where } n \text{ is the product of } n_0 \text{ and } n_1$$

e.g. $\langle a * b,\ (a = 5, b = 20)\rangle \equiv 100$

Example arith. expr. evaluation

Evaluating meaning of a complicated arith. expr. will require

- several application of the above rules
- operator precedence

$$\frac{\dfrac{\langle a,\ (a = 5, b = 20)\rangle \equiv 5 \quad \langle b,\ (a = 5, b = 20)\rangle \equiv 20}{\langle a * b,\ (a = 5, b = 20)\rangle \equiv 100} \quad \langle b,\ (a = 5, b = 20)\rangle \equiv 20}{\langle a * b + b,\ (a = 5, b = 20)\rangle \equiv 120}$$

Meaning of Boolean Expressions B - (1)

- $\langle true, s\rangle \equiv true$
- $\langle false, s\rangle \equiv false$
- Equality Check

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 = a_1, s\rangle \equiv true} \text{ where } n_0 \text{ and } n_1 \text{ are equal}$$

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 = a_1, s\rangle \equiv false} \text{ where } n_0 \text{ and } n_1 \text{ are unequal}$$

Meaning of Boolean Expressions B - (2)

- LEQ check

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 \leq a_1, s\rangle \equiv true} \text{ where } n_0 \text{ is l.e.q. to } n_1$$

$$\frac{\langle a_0, s\rangle \equiv n_0 \quad \langle a_1, s\rangle \equiv n_1}{\langle a_0 \leq a_1, s\rangle \equiv false} \text{ where } n_0 \text{ is greater than } n_1$$

Meaning of Boolean Expressions B - (3)

- Negation

$$\frac{\langle b, s\rangle \equiv true}{\langle \neg b, s\rangle \equiv false} \quad \frac{\langle b, s\rangle \equiv false}{\langle \neg b, s\rangle \equiv true}$$

- Conjunction

$$\frac{\langle b_0, s\rangle \equiv t_0 \quad \langle b_1, s\rangle \equiv t_1}{\langle b_0 \wedge b_1, s\rangle \equiv t} \text{ where } t \text{ is logical conjunction of } t_0,\ t_1$$

Example of Boolean Expr. meaning

$$\frac{\dfrac{\langle a,\ (a=5, b=6)\rangle \equiv 5 \quad \langle b,\ (a=5, b=6)\rangle \equiv 6}{\langle a=b,\ (a=5, b=6)\rangle \equiv false}}{\langle \neg a=b,\ (a=5, b=6)\rangle \equiv true}$$

$$\frac{\dfrac{\langle a,\ s\rangle \equiv 5 \quad \langle b,\ s\rangle \equiv 6}{\langle a \le b,\ s\rangle \equiv true} \qquad \dfrac{\langle a,\ s\rangle \equiv 5 \quad \langle b,\ s\rangle \equiv 6}{\langle a=b,\ s\rangle \equiv false}}{\langle a \le b \wedge a=b,\ s\rangle \equiv false}$$

where s is the state $(a=5, b=6)$

Meaning of Expressions

- Expressions evaluate to values in a given state.
- Therefore, the meaning of expressions are given by values.
  - boolean values for boolean expressions
  - numbers for arithmetic expressions
- Using the meaning of expressions, we can assign meaning to commands.

Meaning of Commands

- Execution of commands leads to a change of program state.
- Therefore the meaning of a command c is: If c is executed in some state s, how does it change s to s'.

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

Rules for commands - (1)

- Skip

$$\langle skip, s\rangle \rightarrow s$$

- Sequencing

$$\frac{\langle c_0, s\rangle \rightarrow s_{int} \quad \langle c_1, s_{int}\rangle \rightarrow s'}{\langle c_0; c_1, s\rangle \rightarrow s'}$$

Rules for commands - (2)

- Assignment

$$\frac{\langle a, s\rangle \equiv n}{\langle X := a, s\rangle \rightarrow s[X=n]}$$

where $s[X=n]$ is a state which is same as state s, except that the value of varible X in $s[X=n]$ is n.

Thus:

$(a=5, b=20, c=2)[a=7]$ is the state $(a=7, b=20, c=2)$

$(a=5, b=20, c=2)[a=5]$ is the state $(a=5, b=20, c=2)$

Rule for Commands (3)

- If-then-else

$$\frac{\langle b, s\rangle \equiv true \quad \langle c_0, s\rangle \rightarrow s'}{\langle if\ b\ then\ c_0\ else\ c_1, s\rangle \rightarrow s'}$$

$$\frac{\langle b, s\rangle \equiv false \quad \langle c_1, s\rangle \rightarrow s'}{\langle if\ b\ then\ c_0\ else\ c_1, s\rangle \rightarrow s'}$$

Rule for Commands (4)

- While

$$\frac{\langle b, s\rangle \equiv false}{\langle while\ b\ do\ c, s\rangle \rightarrow s}$$

$$\frac{\langle b, s\rangle \equiv true \quad \langle c, s\rangle \rightarrow s_{int} \quad \langle while\ b\ do\ c, s_{int}\rangle \rightarrow s'}{\langle while\ b\ do\ c, s\rangle \rightarrow s'}$$

Summary of rules

- The meaning of each commands specifies how an execution of the command chnages state.
- Roughly speaking, this is done by simulating the execution of the commands.
- For example, the rule for while essentially unfolds the iterations of the while loop.