The While programming language

Matthew Hennessy

October 11, 2010

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

The language *While*

$$
\begin{array}{rcl}
B \in Bool & ::= & \texttt{true} \mid \texttt{false} \mid E = E \mid B \& B \mid \neg\, B \mid \cdots \\
E \in Exp & ::= & \textsc{l} \mid \texttt{n} \mid (E + E) \mid \cdots \\
C \in Com & ::= & \textsc{l} := E \mid \texttt{if } B \texttt{ then } C \texttt{ else } C \\
 & \mid & C \,;\, C \mid \texttt{skip} \mid \texttt{while } B \texttt{ do } C
\end{array}
$$

$\textsc{l}$, $\textsc{k}$ from a set of locations $\textsc{Locs}$

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Example program

$$
\begin{array}{l}
\mathrm{L}_2 := 1; \\
\mathrm{L}_3 := 0; \\
\texttt{while} \neg (\mathrm{L}_1 = \mathrm{L}_2) \texttt{ do} \\
\quad \mathrm{L}_2 := \mathrm{L}_2 + 1; \\
\quad \mathrm{L}_3 := \mathrm{L}_3 + 1;
\end{array}
$$

How do we describe the behaviour of these programs?

How can we prescribe how these programs should be executed?

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

Dependencies

Behaviour of commands in *Com* depend on behaviour of Boolean expressions in *Bool*

- $\texttt{if } \mathrm{L} = \mathrm{K} \texttt{ then } \mathrm{L}_1 := \mathrm{K} \texttt{ else } \mathrm{L}_2 := \mathrm{L}$
- $\texttt{while } \neg (\mathrm{L}_1 = \mathrm{L}_2) \texttt{ do } \mathrm{L}_2 := \mathrm{L}_2 + 1 \,;\, \mathrm{L}_3 := \mathrm{L}_3 + 1$

Behaviour of Boolean in *Bool* depend on behaviour of expressions in *Exp*

- $(\mathrm{L} + 1) \; = (\mathrm{K} + 2)$
- $(\mathrm{L}_2 + \mathrm{L}) \; = \; \mathrm{K}$

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

Evaluating expressions

Value of expressions depend on current values in locations

- $\mathrm{K} + \mathrm{L} - 1$

Value depends on current values of locations K and L

Values stored at locations change as programs are executed

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

States

- A *state* (of the memory) is a function from locations to numerals, $s : \textsc{Locs} \rightarrow \mathit{Nums}$.
- The state $s[\mathrm{K} \mapsto n]$ is defined by

$$s[\mathrm{K} \mapsto n](\mathrm{L}) = \begin{cases} \mathtt{n} & \text{if } \mathrm{K} = \mathrm{L} \\ s(\mathrm{L}) & \text{otherwise} \end{cases}$$

- Behaviour of commands is relative to a *state*
- The state changes as the execution of a command proceeds
- Complete execution of a command transforms an initial state into a terminal state

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

Big-step semantics of arithmetic expressions

Judgements:

$$\langle E, s \rangle \Downarrow \mathtt{n}$$

meaning: value of expression E relative to the state s is $\mathtt{n}$

Alternative:

$$\langle E, s \rangle \Downarrow \langle \mathtt{n}, s' \rangle$$

meaning:

- E relative to the state s evaluates to $\mathtt{n}$
- the evaluation changes the state from s to s'

Big-step semantics of arithmetics

(B-NUM)
$$\frac{}{\langle \mathtt{n}, s \rangle \Downarrow \mathtt{n}}$$

(B-ADD)
$$\frac{\langle E_1, s \rangle \Downarrow \mathtt{n}_1 \qquad \langle E_2, s \rangle \Downarrow \mathtt{n}_2}{\langle E_1 + E_2, s \rangle \Downarrow \mathtt{n}_3} \quad n_3 = \mathsf{add}(n_1, n_2)$$

(B-LOC)
$$\frac{}{\langle \mathrm{L}, s \rangle \Downarrow s(\mathrm{L})}$$

Assignments

Evaluate the command $(\mathrm{L} := E)$ *relative to state* s*?*

Intuition:

(1) evaluate E relative to state s to some value $\mathtt{n}$

(2) update location L with new value $\mathtt{n}$

Inference rule:

$$\text{(B-ASSIGN.S)}\quad \frac{\langle E, s\rangle \Downarrow \mathtt{n}}{\langle \mathrm{L} := E, s\rangle \Downarrow s[\mathrm{L} \mapsto \mathtt{n}]}$$

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

Sequential composition

Evaluate command $C_1 ; C_2$ *relative to state* s*?*

Intuition:

(1) evaluate C_1 relative to state s, to get new state s_1

(2) then evaluate C_2 relative to new state s_1

Rule:

$$\text{(B-SEQ.S)}\quad \frac{\langle C_1, s\rangle \Downarrow s_1 \qquad \langle C_2, s_1\rangle \Downarrow s'}{\langle C_1 ; C_2, s\rangle \Downarrow s'}$$

DUBLIN
Átha Cliath

If commands

Evaluate command (`if` B `then` C_1 `else` C_2) *relative to state* s?

Intuition:

(1) first evaluate B to some boolean value `bv`

(2) if `true` evaluate C_1 relative to state s

(3) if `false` evaluate C_2 relative to state s

Rules:

(B-IF.T)
$$\frac{\langle B, s\rangle \Downarrow \texttt{true} \qquad \langle C_1, s\rangle \Downarrow s'}{\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \Downarrow s'}$$

(B-IF.F)
$$\frac{\langle B, s\rangle \Downarrow \texttt{false} \qquad \langle C_2, s\rangle \Downarrow s'}{\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \Downarrow s'}$$

DUBLIN
ÁTHA CLIATH

While commands

Evaluate command (`while` B `do` C) *relative to state* s?

Intuition:

(1) first evaluate B to some boolean value `bv`

(2) if `false` nothing to be done

(3) if `true` evaluate C relative to state s to get new state s_1

(4) then evaluate original (`while` B `do` C) relative to s_1

Rules:

(B-WHILE.F)
$$\frac{\langle B, s\rangle \Downarrow \texttt{false}}{\langle \texttt{while } B \texttt{ do } C, s\rangle \Downarrow s}$$

(B-WHILE.T)
$$\frac{\langle B, s\rangle \Downarrow \texttt{true} \qquad \langle C, s\rangle \Downarrow s_1 \qquad \langle \texttt{while } B \texttt{ do } C, s_1\rangle \Downarrow s'}{\langle \texttt{while } B \texttt{ do } C, s\rangle \Downarrow s'}$$

DUBLIN
ÁTHA CLIATH

The skip command

Evaluate command `skip` *relative to state s?*

Intuition:

(1) nothing to do

Rule:

$$\frac{}{\langle \mathtt{skip}, s\rangle \Downarrow s} \;(\textsc{b-skip})$$

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Properties of big-step semantics

Strong normalisation:

For every state s and every command C there exists some state s' such that $\langle C, s\rangle \Downarrow s'$ **False**

Determinacy:

If $\langle C, s\rangle \Downarrow s_1$ and $\langle C, s\rangle \Downarrow s_2$ then $s_1 = s_2$ **True**

Proof requires rule induction

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Non-termination in big-step semantics

- Let C be $\texttt{while} \neg (\mathrm{L} = 0)$ do $\mathrm{L} := \mathrm{L} + 1$
- Let $s(\mathrm{L}) = 1$
- How can we derive $\langle C, s \rangle \Downarrow s'$ for any s'?
- What is the shortest proof of judgement of the form $\langle C, s \rangle \Downarrow s'$?

The meaning of commands

$$
\begin{array}{l}
\mathrm{L}_2 := 1; \\
\mathrm{L}_3 := 0; \\
\texttt{while} \neg (\mathrm{L}_1 = \mathrm{L}_2) \texttt{ do} \\
\quad \mathrm{L}_2 := \mathrm{L}_2 + 1; \\
\quad \mathrm{L}_3 := \mathrm{L}_3 + 1; \\
\mathrm{L}_1 := \mathrm{L}_3
\end{array}
$$

What does this program do?

- A program transforms an initial state in a terminal state
- For some initial states there may be no terminal state

Partial functions

$$f : A \rightharpoonup B$$

Meaning:

f calculates an element of B for some elements of A

Notation:

- A is the domain of f
- B is the range of f

Note: $f(a)$ may not be defined for some a in A

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

The meaning of commands

$$[\![-]\!] : Com \longrightarrow (States \rightharpoonup States)$$

$[\![C]\!]$ *transforms an initial state* s *into a terminal state*

Definition:

$$[\![C]\!](s) = \begin{cases} s', & \text{if } \langle C, s \rangle \Downarrow s' \\ \text{undefined}, & \text{otherwise} \end{cases}$$

Determinacy ensures this is a proper definition

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Example

Let C denote
$$\begin{array}{l} \mathrm{L}_2 := 1; \\ \mathrm{L}_3 := 0; \\ \texttt{while} \neg (\mathrm{L}_1 = \mathrm{L}_2) \ \texttt{do} \\ \quad \mathrm{L}_2 := \mathrm{L}_2 + 1; \\ \quad \mathrm{L}_3 := \mathrm{L}_3 + 1; \\ \mathrm{L}_1 := \mathrm{L}_3 \end{array}$$

How do we describe $[\![C]\!]$?

$[\![C]\!](s)$ is a state:

$$[\![C]\!](s)(\mathrm{L}) = \begin{cases} s(\mathrm{L}_1) - 1, & \text{if } s(\mathrm{L}_1) > 0, \mathrm{L} = \mathrm{L}_1, \mathrm{L}_2, \mathrm{L}_3 \\ s(\mathrm{L}), & \text{otherwise} \end{cases}$$

DUBLIN
ÁTHA CLIATH

Small-step semantics for *While*

Judgements:

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

Meaning:

- starting from state s
- when executing command C

one step of computation leads to

- state s'
- with command C' remaining to be executed

What is a step?

Depends

DUBLIN
ÁTHA CLIATH

What is in a step?

Decision:

- Ignore how expressions, Booleans, are evaluated
- One step consists of:
  - memory update
  - or branching decision

Concentrate on execution of commands

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Assignment

How to execute one step of command $(\mathrm{L} := E)$ relative to the state s ?

Intuition:

- Evaluate E relative to state s
- Update state s with resulting value

Inference rule:

$$\text{(W-ASS)}\quad \frac{\langle E, s\rangle \Downarrow \mathtt{n}}{\langle \mathrm{L} := E, s\rangle \longrightarrow \langle \mathtt{skip}, s[\mathrm{L} \mapsto \mathtt{n}]\rangle}$$

One step suffices for entire execution – ignoring evaluation of E

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Conditional

How to execute one step of (**if** B **then** C_1 **else** C_2) relative to state s

Intuition:

- Evaluate B relative to state s
- If `true` start evaluating command C_1
- If `false` start evaluating command C_2

Inference rule:

(W-COND.TRUE)
$$\frac{\langle B, s\rangle \Downarrow \mathtt{true}}{\langle \mathtt{if}\ B\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2, s\rangle \rightarrow \langle C_1, s\rangle}$$

(W-COND.FALSE)
$$\frac{\langle B, s\rangle \Downarrow \mathtt{false}}{\langle \mathtt{if}\ B\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2, s\rangle \rightarrow \langle C_2, s\rangle}$$

DUBLIN
ÁTHA CLIATH

Sequential composition

How to execute one step of command (C_1 ; C_2) relative to state s

Intuition:

- Execute one step of C_1 relative to state s
- If C_1 has terminated start executing C_2

`skip` indicates termination

Inference rule:

(W-SEQ.LEFT)
$$\frac{\langle C_1, s\rangle \rightarrow \langle C_1', s'\rangle}{\langle C_1 \,;\, C_2, s\rangle \rightarrow \langle C_1' \,;\, C_2, s'\rangle}$$

(W-SEQ.SKIP)
$$\frac{}{\langle \mathtt{skip} \,;\, C_2, s\rangle \rightarrow \langle C_2, s\rangle}$$

DUBLIN
ÁTHA CLIATH

While commands

How to execute one step of command (`while` B `do` C) relative to state s

Intuition:

- Evaluate B relative to s
- If `false` then terminate
- if `true` then execute one step of C ......

Inference rule:

(W-WHILE.FALSE)
$$\frac{\langle B, s\rangle \Downarrow \texttt{false}}{\langle \texttt{while } B \texttt{ do } C, s\rangle \rightarrow \langle \texttt{skip}, s\rangle}$$

(W-WHILE.TRUE)
$$\frac{\langle B, s\rangle \Downarrow \texttt{true}}{\langle \texttt{while } B \texttt{ do } C, s\rangle \rightarrow \langle C \,;\, \texttt{while } B \texttt{ do } C, s\rangle}$$

DUBLIN
ÁTHA CLIATH

While loops: the unwinding rule

How to execute one step of command (`while` B `do` C) relative to state s

Intuition:

- combination of (`if` B `then` C `else` ...) and sequential composition

Inference rule:

(W-WHILE)
$$\frac{}{\begin{array}{l}\langle \texttt{while } B \texttt{ do } C, s\rangle \rightarrow \\ \langle \texttt{if } B \texttt{ then } (C \,;\, \texttt{while } B \texttt{ do } C) \texttt{ else skip}, s\rangle\end{array}}$$

DUBLIN
ÁTHA CLIATH

Running commands

To run command P from state s:

Find state s' such that $\langle P, s\rangle \rightarrow^* \langle \texttt{skip}, s'\rangle$

Example:

See McGusker notes, slide 50

Configurations $\langle \texttt{skip}, s\rangle$ are **terminal**

Running commands: Problems can occur

Infinite loops:

Let P be command `while true do skip`

- $\langle P, s\rangle \rightarrow^3 \langle P, s\rangle \rightarrow^3 \langle P, s\rangle \rightarrow^3 \langle P, s\rangle \rightarrow \ldots$
- No state s' such that $P \rightarrow^* \langle \texttt{skip}, s'\rangle$

Progress property:

- Configurations $\langle \texttt{skip}, s\rangle$ are **terminal**
- Either $\langle C, s\rangle$ is terminal or $\langle C, s\rangle \rightarrow \langle C', s'\rangle$ for some configuration $\langle C', s'\rangle$

Questions Questions Questions

- Determinacy:
  - $\langle C, s\rangle \longrightarrow^* s_1$ and $\langle C, s\rangle \longrightarrow^* s_2$ implies $s_1 = s_2$?
- Consistency with big-step semantics:
  - $\langle C, s\rangle \Downarrow s'$ implies $\langle C, s\rangle \longrightarrow^* s'$?
  - $\langle C, s\rangle \longrightarrow^* s'$ implies $\langle C, s\rangle \Downarrow s'$?

Proof strategy:

Similiar to that used for expression language *Exp*

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath