The While programming language

Matthew Hennessy

January 28, 2015

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

The language *While*

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

L, K from a set of locations 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?

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 *Arith*

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

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, $s : \mathrm{LOCS} \to \mathit{Nums}$.
- 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:

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

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:

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

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)
$$\frac{\langle B, s\rangle \Downarrow \texttt{true} \quad \langle C_1, s\rangle \Downarrow s'}{\langle \texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2, s\rangle \Downarrow s'}$$

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

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)
$$\frac{\langle B, s\rangle \Downarrow \texttt{true} \quad \langle C, s\rangle \Downarrow s_1 \quad \langle \texttt{while } B \texttt{ do } C, s_1\rangle \Downarrow s_2}{\langle \texttt{while } B \texttt{ do } C, s\rangle \Downarrow s_2}$$

DUBLIN ÁTHA CLIATH

The `skip` command

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

Intuition:

(1) nothing to do

Rule:

$$\frac{}{\langle \texttt{skip}, s\rangle \Downarrow s} \text{(B-SKIP)}$$

Properties of big-step semantics

Normalisation:

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

Determinacy:

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

Proof requires rule induction

Non-termination in big-step semantics

- Let C be $\texttt{while}\ \neg\ (\mathrm{L} = 0)\ \texttt{do}\ \mathrm{L} := \mathrm{L} + 1$
- Let $s(\mathrm{L}) = 1$

- How can we derive $\langle C, s\rangle \Downarrow s'$ for any s' when $s(\mathrm{L}) > 0$?
- What is the shortest proof of judgement of the form $\langle C, s\rangle \Downarrow s'$? where $s(\mathrm{L}) > 0$


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

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


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

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

$$[\![-]\!] : \mathit{Com} \to (\mathit{States} \rightharpoonup \mathit{States})$$

$[\![C]\!]$ *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
$$\begin{aligned}&\mathrm{L}_2 := 1;\\ &\mathrm{L}_3 := 0;\\ &\texttt{while } \neg\,(\mathrm{L}_1 = \mathrm{L}_2) \texttt{ do}\\ &\quad \mathrm{L}_2 := \mathrm{L}_2 + 1;\\ &\quad \mathrm{L}_3 := \mathrm{L}_3 + 1;\\ &\mathrm{L}_1 := \mathrm{L}_3\end{aligned}$$

How do we describe $[\![C]\!] : (\mathit{States} \rightharpoonup \mathit{States})$?

$[\![C]\!](s)$ is only defined when $s(\mathrm{L}_1) > 0$:

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

DUBLIN
ÁTHA CLIATH

Small-step semantics for *While*

Judgements:

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

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

Terminal configurations:

- $\langle \texttt{skip}, s \rangle$ is **terminal** for every s
- $\langle \texttt{skip}, s \rangle \to \langle C, s' \rangle$ **not possible**

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:

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

One step suffices for entire execution – ignoring evaluation of E

Conditional

How to execute one step of $(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ 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:

$$\frac{\text{(B-COND.T)}\quad \langle B, s\rangle \Downarrow \texttt{true}}{\langle \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, s\rangle \to \langle C_1, s\rangle}$$

$$\frac{\text{(B-COND.F)}\quad \langle B, s\rangle \Downarrow \texttt{false}}{\langle \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2, s\rangle \to \langle C_2, s\rangle}$$

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:

$$\frac{\text{(B-SEQ.LEFT)}\quad \langle C_1, s\rangle \to \langle C_1', s'\rangle}{\langle C_1\ ;\ C_2, s\rangle \to \langle C_1'\ ;\ C_2, s'\rangle}$$

$$\frac{\text{(B-SEQ.SKIP)}}{\langle \texttt{skip}\ ;\ C_2, s\rangle \to \langle C_2, s\rangle}$$

While commands

How to execute one step of command $(\texttt{while}\ B\ \texttt{do}\ C)$ relative to state s

Intuition:

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

Inference rule:

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

$$\frac{\langle B, s\rangle \Downarrow \texttt{true}}{\langle \texttt{while}\ B\ \texttt{do}\ C, s\rangle \to \langle C\ ;\texttt{while}\ B\ \texttt{do}\ C, s\rangle}\ (\text{B-WHILE.T})$$

While loops: alternative the unwinding rule

How to execute one step of command $(\texttt{while}\ B\ \texttt{do}\ C)$ relative to state s

Intuition:

- combination of $(\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ \ldots)$ and sequential composition

Inference rule:

$$\frac{}{\langle \texttt{while}\ B\ \texttt{do}\ C, s\rangle \to \langle \texttt{if}\ B\ \texttt{then}\ (C\ ;\texttt{while}\ B\ \texttt{do}\ C)\ \texttt{else}\ \texttt{skip}, s\rangle}\ (\text{B-WHILE})$$

Running commands

To run command C from state s:

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

Example:

- See Course Notes page 49
- See McGusker notes, slide 50

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

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Running commands: Problems can occur

Infinite loops:

Let C be command `while true do skip`

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

Progress property:

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

TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

Questions Questions Questions

- Determinacy:
  - $\langle C, s\rangle \to^* \langle \texttt{skip}, s_1\rangle$ and $\langle C, s\rangle \to^* \langle \texttt{skip}, s_2\rangle$ implies $s_1 = s_2$?

- Consistency with big-step semantics:
  - $\langle C, s\rangle \Downarrow s'$ implies $\langle C, s\rangle \to^* \langle \texttt{skip}, s'\rangle$?
  - $\langle C, s\rangle \to^* \langle \texttt{skip}, s'\rangle$ implies $\langle C, s\rangle \Downarrow s'$?

Proof strategy:

Similiar to that used for expression language *Exp*
More powerful proof principle required

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath