Lecture Notes on

Semantics of Programming Languages

for Part IB of the Computer Science Tripos

Dr Andrew M. Pitts
Cambridge University Computer Laboratory

- $n \in \mathbb{Z} \stackrel{\text{def}}{=} \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, the set of *integers*;
- $b \in \mathbb{B} \stackrel{\text{def}}{=} \{\mathbf{true}, \mathbf{false}\}$, the set of *booleans*;
- $\ell \in \mathbb{L} \stackrel{\text{def}}{=} \{\ell_0, \ell_1, \ell_2, \ell_3, \ldots\}$ a fixed, infinite set of symbols whose elements we will call *locations* (the term *program variable* is also commonly used), because they denote locations for storing integers—the integer expression $!\ell$ denotes the integer currently stored in ℓ;
- $iop \in \mathbb{I}op \stackrel{\text{def}}{=} \{+, -, *, \ldots\}$ a fixed, finite set of integer-valued binary operations;
- $bop \in \mathbb{B}op \stackrel{\text{def}}{=} \{=, <, >, \ldots\}$ a fixed, finite set of boolean-valued binary operations.

LC Syntax

Phrases

$$P ::= C \mid E \mid B$$

Commands

$$\begin{array}{rcl} C & ::= & \mathbf{skip} \mid \ell := E \mid C\,;C \\ & \mid & \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C \mid \mathbf{while}\ B\ \mathbf{do}\ C \end{array}$$

Integer expressions

$$E ::= n \mid !\ell \mid E\ iop\ E$$

Boolean expressions

$$B ::= b \mid E\ bop\ E$$

Transition systems defined

A *transition system* is specified by

- a set $Config$, and
- a binary relation $\rightarrow \subseteq Config \times Config$.

The elements of $Config$ are often called *configurations* (or 'states'), and the binary relation is written infix, i.e. $\boxed{c \rightarrow c'}$ means c and c' are related by $\rightarrow$.

LC transition relation — expressions

$(\overrightarrow{\text{loc}})$ $\quad \langle !\ell, s\rangle \to \langle n, s\rangle$ if $\ell \in dom(s)$ & $s(\ell) = n$

$$(\overrightarrow{\text{op1}}) \quad \frac{\langle E_1, s\rangle \to \langle E_1', s'\rangle}{\langle E_1\ op\ E_2, s\rangle \to \langle E_1'\ op\ E_2, s'\rangle}$$

$$(\overrightarrow{\text{op2}}) \quad \frac{\langle E_2, s\rangle \to \langle E_2', s'\rangle}{\langle n_1\ op\ E_2, s\rangle \to \langle n_1\ op\ E_2', s'\rangle}$$

$(\overrightarrow{\text{op3}})$ $\quad \langle n_1\ op\ n_2, s\rangle \to \langle c, s\rangle$ if $c = n_1\ op\ n_2$

LC transition relation — := and ;

$$(\overrightarrow{\text{set1}}) \quad \frac{\langle E, s\rangle \to \langle E', s'\rangle}{\langle \ell := E, s\rangle \to \langle \ell := E', s'\rangle}$$

$(\overrightarrow{\text{set2}})$ $\quad \langle \ell := n, s\rangle \to \langle \mathbf{skip}, s[\ell \mapsto n]\rangle$

$$(\overrightarrow{\text{seq1}}) \quad \frac{\langle C_1, s\rangle \to \langle C_1', s'\rangle}{\langle C_1\ ;\ C_2, s\rangle \to \langle C_1'\ ;\ C_2, s'\rangle}$$

$(\overrightarrow{\text{seq2}})$ $\quad \langle \mathbf{skip}\ ;\ C, s\rangle \to \langle C, s\rangle$

LC transition relation — conditional & while

$$(\overrightarrow{\text{if1}}) \quad \frac{\langle B, s\rangle \to \langle B', s'\rangle}{\langle \mathbf{if}\ B\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, s\rangle \to \langle \mathbf{if}\ B'\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2, s'\rangle}$$

$(\overrightarrow{\text{if2}})$ $\quad \langle \mathbf{if\ true\ then}\ C_1\ \mathbf{else}\ C_2, s\rangle \to \langle C_1, s\rangle$

$(\overrightarrow{\text{if3}})$ $\quad \langle \mathbf{if\ false\ then}\ C_1\ \mathbf{else}\ C_2, s\rangle \to \langle C_2, s\rangle$

$(\overrightarrow{\text{whl}})$ $\quad \langle \mathbf{while}\ B\ \mathbf{do}\ C, s\rangle \to \langle \mathbf{if}\ B\ \mathbf{then}\ (C\ ;\ \mathbf{while}\ B\ \mathbf{do}\ C)\ \mathbf{else\ skip}, s\rangle$

Terminal and stuck LC configurations

The *terminal* configurations are by definition

$$\langle n, s\rangle \quad \langle \mathbf{true}, s\rangle \quad \langle \mathbf{false}, s\rangle \quad \langle \mathbf{skip}, s\rangle.$$

A configuration $\langle P, s\rangle$ is *stuck* if and only if it is not terminal, but $\langle P, s\rangle \not\to$.

(For example, $\langle (!\ell + 1), \{\ell' \mapsto 1\}\rangle$ is stuck if $\ell' \neq \ell$.)

$$\langle C, s\rangle \to \langle \textbf{if } B \textbf{ then } (C' ; C) \textbf{ else skip}, s\rangle$$
$$\to \langle \textbf{if } 4 > 0 \textbf{ then } (C' ; C) \textbf{ else skip}, s\rangle$$
$$\to \langle \textbf{if true then } (C' ; C) \textbf{ else skip}, s\rangle$$
$$\to \langle C' ; C, s\rangle$$
$$\to^* \langle \textbf{skip}, s[\ell \mapsto 0, \ell' \mapsto 24]\rangle$$

---

where $\begin{cases} C & \stackrel{\text{def}}{=} \textbf{while } B \textbf{ do } C', \\ B & \stackrel{\text{def}}{=} !\ell > 0, \\ C' & \stackrel{\text{def}}{=} \ell' := !\ell * !\ell' ; \ell := !\ell - 1, \\ s & \stackrel{\text{def}}{=} \{\ell \mapsto 4, \ell' \mapsto 1\}. \end{cases}$

$(\Downarrow_{\text{con}})$ $\langle c, s\rangle \Downarrow \langle c, s\rangle \quad (c \in \mathbb{Z} \cup \mathbb{B})$

$(\Downarrow_{\text{loc}})$ $\langle !\ell, s\rangle \Downarrow \langle n, s\rangle$ if $\ell \in dom(s)$ & $s(\ell) = n$

$(\Downarrow_{\text{op}})$ $\dfrac{\langle E_1, s\rangle \Downarrow \langle n_1, s'\rangle \quad \langle E_2, s'\rangle \Downarrow \langle n_2, s''\rangle}{\langle E_1 \; op \; E_2, s\rangle \Downarrow \langle c, s''\rangle}$ where c is the value of $n_1 \; op \; n_2$ (for op an integer or boolean operation)

$(\Downarrow_{\text{skip}})$ $\langle \textbf{skip}, s\rangle \Downarrow \langle \textbf{skip}, s\rangle$

$(\Downarrow_{\text{set}})$ $\dfrac{\langle E, s\rangle \Downarrow \langle n, s'\rangle}{\langle \ell := E, s\rangle \Downarrow \langle \textbf{skip}, s'[\ell \mapsto n]\rangle}$

$(\Downarrow_{\text{seq}})$ $\dfrac{\langle C_1, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle}{\langle C_1 ; C_2, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle}$

$(\Downarrow_{\text{if1}})$ $\dfrac{\langle B, s\rangle \Downarrow \langle \textbf{true}, s'\rangle \quad \langle C_1, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle}{\langle \textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle}$

$(\Downarrow_{\text{if2}})$ $\dfrac{\langle B, s\rangle \Downarrow \langle \textbf{false}, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle}{\langle \textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle}$

plus rules $(\Downarrow_{\text{wh1}})$ and $(\Downarrow_{\text{wh2}})$ on Slide 26.

Evaluation rules for while

$(\Downarrow_{\text{wh1}})$ $\dfrac{\langle B, s\rangle \Downarrow \langle \textbf{true}, s'\rangle \quad \langle C, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle \quad \langle \textbf{while } B \textbf{ do } C, s''\rangle \Downarrow \langle \textbf{skip}, s'''\rangle}{\langle \textbf{while } B \textbf{ do } C, s\rangle \Downarrow \langle \textbf{skip}, s'''\rangle}$

$(\Downarrow_{\text{wh2}})$ $\dfrac{\langle B, s\rangle \Downarrow \langle \textbf{false}, s'\rangle}{\langle \textbf{while } B \textbf{ do } C, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle}$

For $\begin{cases} C & \stackrel{\text{def}}{=} \textbf{while } !\ell > 0 \textbf{ do } \ell := 0 \\ s & \stackrel{\text{def}}{=} \{\ell \mapsto 1\} \end{cases}$

we try to find s'' such that $\langle C, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle$ is provable. Since $\langle !\ell > 0, s\rangle \Downarrow \langle \textbf{true}, s\rangle$ (proof shown below), the last rule used in the proof must be $(\Downarrow_{\text{wh1}})$:

$$\dfrac{\dfrac{}{\langle !\ell, s\rangle \Downarrow \langle 1, s\rangle}(\Downarrow_{\text{loc}}) \quad \dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}(\Downarrow_{\text{con}})}{\langle !\ell > 0, s\rangle \Downarrow \langle \textbf{true}, s\rangle}(\Downarrow_{\text{op}}) \quad \begin{matrix}?\\ \vdots \\ \langle \ell := 0, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle\end{matrix} \quad \begin{matrix}?\\ \vdots \\ \langle C, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle\end{matrix}}{\langle C, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle}(\Downarrow_{\text{wh1}})$$

for some s' and s''. The middle hypothesis of $(\Downarrow_{\text{wh1}})$ must have been deduced using $(\Downarrow_{\text{set}})$. So $s' = \{\ell \mapsto 0\}$ and we have:

$$\dfrac{\dfrac{\dfrac{}{\langle !\ell, s\rangle \Downarrow \langle 1, s\rangle}(\Downarrow_{\text{loc}}) \quad \dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}(\Downarrow_{\text{con}})}{\langle !\ell > 0, s\rangle \Downarrow \langle \textbf{true}, s\rangle}(\Downarrow_{\text{op}}) \quad \dfrac{\dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}(\Downarrow_{\text{con}})}{\langle \ell := 0, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle}(\Downarrow_{\text{set}}) \quad \begin{matrix}?\\ \vdots \\ \langle C, s'\rangle \Downarrow \langle \textbf{skip}, s''\rangle\end{matrix}}{\langle C, s\rangle \Downarrow \langle \textbf{skip}, s''\rangle}(\Downarrow_{\text{wh1}}).$$

Finally, since $\langle !\ell > 0, s'\rangle \Downarrow \langle \textbf{false}, s'\rangle$ (proof shown below), the last rule used in the proof of the right-hand branch must be $(\Downarrow_{\text{wh2}})$. So $s'' = s' = \{\ell \mapsto 0\}$ and the complete proof is:

$$\dfrac{\dfrac{\dfrac{}{\langle !\ell, s\rangle \Downarrow \langle 1, s\rangle}(\Downarrow_{\text{loc}}) \quad \dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}(\Downarrow_{\text{con}})}{\langle !\ell > 0, s\rangle \Downarrow \langle \textbf{true}, s\rangle}(\Downarrow_{\text{op}}) \quad \dfrac{\dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}(\Downarrow_{\text{con}})}{\langle \ell := 0, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle}(\Downarrow_{\text{set}}) \quad \dfrac{\dfrac{\dfrac{}{\langle !\ell, s'\rangle \Downarrow \langle 0, s'\rangle}(\Downarrow_{\text{loc}}) \quad \dfrac{}{\langle 0, s'\rangle \Downarrow \langle 0, s'\rangle}(\Downarrow_{\text{con}})}{\langle !\ell > 0, s'\rangle \Downarrow \langle \textbf{false}, s'\rangle}(\Downarrow_{\text{op}})}{\langle C, s'\rangle \Downarrow \langle \textbf{skip}, s'\rangle}(\Downarrow_{\text{wh2}})}{\langle C, s\rangle \Downarrow \langle \textbf{skip}, s'\rangle}(\Downarrow_{\text{wh1}}).$$

型に対する単一化アルゴリズム

1 型構成子 (type constructor)

型構成子は，0 個以上の型から新たな型を構成するための記法である．基本型 (bool, int 等)，list，関数型 ($\rightarrow$) はそれぞれ，0 引数，1 引数，2 引数の型構成子と考えることができる．以下では型構成子を一般に F, G などと書く．$t_1 \rightarrow t_2$ は，$\rightarrow(t_1, t_2)$ の略記と考える．

2 単一化 (unification) アルゴリズム

単一化とは，二つの (型変数を含むかもしれない) 型を同じ形の型にするための代入 (= 型変数から型への写像) を求める作業である．二つの型 s と t を単一化するアルゴリズムを以下に示す．

1. $S \leftarrow \{s = t\}$
2. 集合 S の中の等式に対して下記の変換を繰り返し行なう．複数の変換ルールが適用可能な場合はどれを先に適用してもよい．(a)–(f) のどの変換も適用不可能になったら成功終了する．
   (a) $F(s_1, \ldots, s_n) = F(t_1, \ldots, t_n)$ $(n \geq 0)$ の形の等式があったら，これを n 本の等式 $s_1 = t_1, \ldots, s_n = t_n$ に置き換える．
   (b) $F(s_1, \ldots, s_m) = G(t_1, \ldots, t_n)$ $(F \neq G$ または $m \neq n)$ の形の等式があったら失敗終了．
   (c) $x = x$ の形の等式 (x は変数) があったらこれを S から消去．
   (d) $t = x$ の形の等式 (t は非変数，x は変数) があったらこれを $x = t$ で置き換える．
   (e) $x = t$ の形の等式 (x は変数，$t \neq x$，x が t 中に現れる) があったら失敗終了．
   (f) $x = t$ の形の等式 (x は変数，$t \neq x$，x は t 中に現れない) があり，かつ x が S 中の他の等式に出現していたら，それらの x の出現を t に書き換える ($x = t$ はそのまま残す)．

定理 このアルゴリズムは必ず成功終了か失敗終了する．成功終了したとき，S には (f) の形の等式だけが残り，$S = \{x_1 = t_1, \ldots, x_n = t_n\}$ $(n \geq 0$, x_i は互いに異なる変数，どの x_i も $t_1, \ldots, t_n$ 中に出現しない) となっているはずである．このとき，代入 $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ が s と t の most general unifier (mgu)，つまり s と t を同じ型にする最も一般的な代入になっている．失敗終了したとき，s と t は単一化不可能である．

3 一般化

一対の型 (s, t) ではなく、いくつかの型の対 $(s_1, t_1), \ldots, (s_n, t_n)$ を同時に単一化することもできる．アルゴリズムのステップ 1. を $S \leftarrow \{s_1 = t_1, \ldots, s_n = t_n\}$ に変更すればよい．

二つの型の join と meet

型階層は，型を要素とする擬順序集合（preordered set，要素間に擬順序関係が導入された集合）である．擬順序集合では，しばしば，二つの要素の共通の祖先や共通の子孫が興味の対象となる．一般に，共通の祖先や子孫は存在するとは限らないし，存在しても，共通の祖先の中で最下位のものや共通の子孫のうち最上位のものが一意に定まるとは限らない．

ところが，講義で紹介したレコード型の（subtype-supertype 関係によって導入される）型階層においては，二つの型の join（$\sqcup$，共通の supertype のうち最下位のもの）および二つの型の meet（$\sqcap$，共通の subtype のうち最上位のもの）が，もし存在するならば一意に定まる．

また，型階層の最上位の型 Top 型の存在を仮定すると，二つの型の join は必ず存在する．

以下に，レコード型を追加した FL の型体系における join と meet の定義を示す．ここでは，list, $\rightarrow$, 基本型 (int, bool) に加えて，Top も型構成子とみなす．また，すべてのレコード型は共通の型構成子 $\{\cdot\}$ をもつものと仮定する．

1 二つの型の join (least upper bound)

1. $\sigma \sqcup \tau = \mathsf{Top}$;　（$\sigma$ と τ が異なる型構成子をもつ場合）
2. $\sigma \sqcup \sigma = \sigma$;　（$\sigma$ は基本型）
3. $\{a_1 : \sigma_1, \ldots, a_n : \sigma_n, b_1 : \rho_1, \ldots, b_m : \rho_n\} \sqcup \{a_1 : \sigma_1, \ldots, a_n : \sigma_n, b'_1 : \rho'_1, \ldots, b'_{m'} : \rho'_{n'}\}$
   $= \{a_1 : \sigma_1 \sqcup \tau_1, \ldots, a_n : \sigma_n \sqcup \tau_n\}$;　（$n \geq 0, m \geq 0, m' \geq 0$; b_i と b'_j は互いに異なる）
4. $\mathsf{list}(\sigma) \sqcup \mathsf{list}(\tau) = \mathsf{list}(\sigma \sqcup \tau)$
5. $(\sigma \rightarrow \sigma') \sqcup (\tau \rightarrow \tau') = (\sigma \sqcap \tau) \rightarrow (\sigma' \sqcup \tau')$

2 二つの型の meet (greatest lower bound)

1. $\sigma \sqcap \sigma = \sigma$;　（$\sigma$ は基本型）
2. $\sigma \sqcap \mathsf{Top} = \mathsf{Top} \sqcap \sigma = \sigma$
3. $\{a_1 : \sigma_1, \ldots, a_n : \sigma_n, b_1 : \rho_1, \ldots, b_m : \rho_n\} \sqcap \{a_1 : \sigma_1, \ldots, a_n : \sigma_n, b'_1 : \rho'_1, \ldots, b'_{m'} : \rho'_{n'}\}$
   $= \{a_1 : \sigma_1 \sqcap \tau_1, \ldots, a_n : \sigma_n \sqcap \tau_n, b_1 : \rho_1, \ldots, b_m : \rho_n, b'_1 : \rho'_1, \ldots, b'_{m'} : \rho'_{n'}\}$;
   （$n \geq 0, m \geq 0, m' \geq 0$; b_i と b'_j は互いに異なる; $\sigma_j \sqcap \tau_j$ $(1 \leq j \leq n)$ が存在する）
4. $\mathsf{list}(\sigma) \sqcap \mathsf{list}(\tau) = \mathsf{list}(\sigma \sqcap \tau)$
5. $(\sigma \rightarrow \sigma') \sqcap (\tau \rightarrow \tau') = (\sigma \sqcup \tau) \rightarrow (\sigma' \sqcap \tau')$