Program Analysis (70020)

Correctness of an Analysis

Herbert Wiklicky

Department of Computing
Imperial College London

herbert@doc.ic.ac.uk
h.wiklicky@imperial.ac.uk

Autumn 2023

Correctness

Questions: Is a program analysis correct? Are the results reflecting what is really happening when the program is run?

In other words: What is the relation between the (concrete) semantics of a program, i.e. the transition relation $\Rightarrow$ and/or its transitive closure $\Rightarrow^*$, and the (solutions to) the program analysis $\mathit{Analysis}_\circ$ and $\mathit{Analysis}_\bullet$.

For example: Is a variable *LV* identifies as 'live' indeed useful, or more importantly, is a 'non-live' variable really 'dead', i.e. is it save to eliminate it (at least locally).

Syntax of WHILE

The labelled syntax of the language WHILE is given by the following **abstract syntax**:

$$
\begin{array}{lcl}
a & ::= & x \mid n \mid a_1 \; op_a \; a_2 \\
b & ::= & true \mid false \mid not \; b \mid b_1 \; op_b \; b_2 \mid a_1 \; op_r \; a_2 \\
S & ::= & [x := a]^\ell \\
 & & \mid [\mathbf{skip}]^\ell \\
 & & \mid S_1 ; S_2 \\
 & & \mid \mathbf{if} \; [b]^\ell \; \mathbf{then} \; S_1 \; \mathbf{else} \; S_2 \\
 & & \mid \mathbf{while} \; [b]^\ell \; \mathbf{do} \; S
\end{array}
$$

Sketches of a Formal Semantics

Memory is modelled by an abstract state, i.e. functions of type

$$\mathbf{State} = \mathbf{Var} \to \mathbf{Z}.$$

For boolean and arithmetic expressions we assume that we know what they "evaluate to" in a state $s \in \mathbf{State}$. Then the semantics for **AExp** is a *total* function

$$[\![\, . \,]\!]_{\mathcal{A}} \, . \quad : \quad \mathbf{AExp} \to \mathbf{State} \to \mathbf{Z}$$

and the semantics of boolean expressions is given by

$$[\![\, . \,]\!]_{\mathcal{B}} \, . \quad : \quad \mathbf{BExp} \to \mathbf{State} \to \{\mathbf{tt}, \mathbf{ff}\}$$

Evaluating Expressions

Let us look at a program with two variables $\textbf{Var} = \{x, y\}$.
Two possible states in this case could be for example:

$$s_0 = [x \mapsto 0, y \mapsto 1] \text{ and } s_1 = [x \mapsto 1, y \mapsto 1]$$

We can evaluate an expression like $x + y \in \textbf{AExp}$:

$$\begin{aligned} [\![x + y]\!]_{\mathcal{A}} s_0 &= 0 + 1 = 1 \\ [\![x + y]\!]_{\mathcal{A}} s_1 &= 1 + 1 = 2 \end{aligned}$$

or a Boolean expression like $x + y \leq 1 \in \textbf{BExp}$:

$$\begin{aligned} [\![x + y \leq 1]\!]_{\mathcal{B}} s_0 &= 1 \leq 1 = \textbf{tt} \\ [\![x + y \leq 1]\!]_{\mathcal{B}} s_1 &= 2 \leq 1 = \textbf{ff} \end{aligned}$$

Execution and Transitions

The configurations describe the current state of the execution.

$$\begin{aligned} \langle S, s \rangle &\quad \ldots \quad S \text{ is to be executed in state } s, \\ s &\quad \ldots \quad \text{a terminal state (i.e. } \langle ., s \rangle). \end{aligned}$$

The transition relation $\Rightarrow$ specify the (possible) computational steps during the execution starting from a certain configuration

$$\langle S, s \rangle \Rightarrow \langle S', s' \rangle$$

and at the end of the computation

$$\langle S, s \rangle \Rightarrow s'$$

Execution Rules (SOS) [Provided in Exam]

(ass)	$\langle [x := a]^\ell, s\rangle \Rightarrow s[x \mapsto [\![a]\!]_{\mathcal{A}} s]$	
(skip)	$\langle [\mathbf{skip}]^\ell, s\rangle \Rightarrow s$	
(sq^1)	$\dfrac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}$	
(sq^T)	$\dfrac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$	
(if^T)	$\langle \mathbf{if}\ [b]^\ell\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle$	if $[\![b]\!]_{\mathcal{B}} s = \mathbf{tt}$
(if^F)	$\langle \mathbf{if}\ [b]^\ell\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle$	if $[\![b]\!]_{\mathcal{B}} s = \mathbf{ff}$
(wh^T)	$\langle \mathbf{while}\ [b]^\ell\ \mathbf{do}\ S, s\rangle \Rightarrow \langle S; \mathbf{while}\ [b]^\ell\ \mathbf{do}\ S, s\rangle$	if $[\![b]\!]_{\mathcal{B}} s = \mathbf{tt}$
(wh^F)	$\langle \mathbf{while}\ [b]^\ell\ \mathbf{do}\ S, s\rangle \Rightarrow s$	if $[\![b]\!]_{\mathcal{B}} s = \mathbf{ff}$

A SOS Example

Consider a (perhaps rather vacuous) program like:

$$S \equiv [\, z := x + y\,]^\ell; \mathbf{while}\ [\mathit{true}]^{\ell'}\ \mathbf{do}\ [\ \mathbf{skip}\ ]^{\ell''}$$

$s_0 = [x \mapsto 0, y \mapsto 1, z \mapsto 0]$ and $s_1 = [x \mapsto 0, y \mapsto 1, z \mapsto 1]$

Then $\langle S, s_0\rangle$ executes as follows:

$$\begin{array}{lll}
\langle S, s_0\rangle & \Rightarrow & \langle \mathbf{while}\ [\mathit{true}]^{\ell'}\ \mathbf{do}\ [\ \mathbf{skip}\ ]^{\ell''}, s_1\rangle \\
& \Rightarrow & \langle [\ \mathbf{skip}\ ]^{\ell''}; \mathbf{while}\ [\mathit{true}]^{\ell'}\ \mathbf{do}\ [\ \mathbf{skip}\ ]^{\ell''}, s_1\rangle \\
& \Rightarrow & \langle \mathbf{while}\ [\mathit{true}]^{\ell'}\ \mathbf{do}\ [\ \mathbf{skip}\ ]^{\ell''}, s_1\rangle \\
& \Rightarrow & \langle [\ \mathbf{skip}\ ]^{\ell''}; \mathbf{while}\ [\mathit{true}]^{\ell'}\ \mathbf{do}\ [\ \mathbf{skip}\ ]^{\ell''}, s_1\rangle \\
& \Rightarrow & \ldots
\end{array}$$

Lemma 1

(i) If $\langle S, s\rangle \Rightarrow s'$ then
$final(S) = \{init(S)\}$.

(ii) If $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ then
$final(S) \supseteq final(S')$.

(iii) If $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ then
$flow(S) \supseteq flow(S')$.

(iv) If $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ then
$blocks(S) \supseteq blocks(S')$.

(v) If $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$ then
S label consistent implies S' label consistent.

Lemma 1 - Proof (i) [Not for Exam]

Proof.
The proof is by induction on the shape of the inference tree.
Consider the only three non-vacuous cases:

(ass): $\langle [\texttt{x:=}a]^\ell, s\rangle \Rightarrow s[x \mapsto [\![a]\!]s]$

$$final([\texttt{x:=}a]^\ell) = \{\ell\} = \{init([\texttt{x:=}a]^\ell)\}.$$

(skip): $\langle [\texttt{skip}]^\ell, s\rangle \Rightarrow s$

$$final([\texttt{skip}]^\ell) = \{\ell\} = \{init([\texttt{skip}]^\ell)\}.$$

(wh^F): $\langle \texttt{while}\ [b]^\ell\ \texttt{do}\ S, s\rangle \Rightarrow s$ with $[\![b]\!] = \textbf{false}$

$$final(\texttt{while}\ [b]^\ell\ \texttt{do}\ S) = \{\ell\} = \{init(\texttt{while}\ [b]^\ell\ \texttt{do}\ S)\}.$$

Lemma 1 - Proof (ii) [Not for Exam]

Proof (cont).

(seq[1]): $\langle S_1\,;S_2, s\rangle \Rightarrow \langle S_1'\,;S_2, s'\rangle$ because $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$:

$$final(S_1\,;S_2) = final(S_2) = final(S_1'\,;S_2).$$

(seq^T): . . .

(if^T): $\langle \texttt{if}\ [b]^\ell\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle$ with $[\![b]\!] = \mathbf{true}$:

$$final(\texttt{if}\ [b]^\ell\ \texttt{then}\ S_1\ \texttt{else}\ S_2) = final(S_1) \cup final(S_2) \supseteq final(S_1).$$

(if^F): . . .

(wh^T): . . .

□

LV Equations: $LV^=$

The Live Variable Analysis is given as the solution to the following system of equations:

$$\mathrm{LV}_{exit}(\ell) = \begin{cases} \emptyset, \text{if } \ell \in final(S_\star) \\ \bigcup\{\mathrm{LV}_{entry}(\ell') \mid (\ell', \ell) \in flow^R(S_\star)\}, \text{otherwise} \end{cases}$$

$$\mathrm{LV}_{entry}(\ell) = (\mathrm{LV}_{exit}(\ell) \backslash kill_{\mathrm{LV}}([B]^\ell) \cup gen_{\mathrm{LV}}([B]^\ell) \quad \text{where } [B]^\ell \in blocks(S_\star)$$

Solutions via Iteration Operator

$$
\begin{array}{rcl}
\mathrm{LV}_{entry}(1) & = & \mathbf{F}^{\bullet}_{1}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n)) \\
\ldots & \ldots & \ldots \\
\mathrm{LV}_{entry}(n) & = & \mathbf{F}^{\bullet}_{n}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n)) \\
\mathrm{LV}_{exit}(1) & = & \mathbf{F}^{\circ}_{1}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n)) \\
\ldots & \ldots & \ldots \\
\mathrm{LV}_{exit}(n) & = & \mathbf{F}^{\circ}_{n}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n))
\end{array}
$$

becomes a function on the lattice $\mathcal{P}(\mathbf{Var})^{2n}$

$$\mathbf{F} : \mathcal{P}(\mathbf{Var})^{2n} \to \mathcal{P}(\mathbf{Var})^{2n}$$

$$
\begin{array}{rcl}
\mathbf{F}^{\bullet}_{i}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n)) & = & \mathrm{LV}_{entry}(i) \\
\mathbf{F}^{\circ}_{i}(\mathrm{LV}_{entry}(1), \ldots, \mathrm{LV}_{exit}(n)) & = & \mathrm{LV}_{exit}(i)
\end{array}
$$

LV Constraints: $LV^{\subseteq}$

The Live Variable Analysis is equivalently given as the solution to the following system of constraints:

$$
\mathrm{LV}_{exit}(\ell) \;\supseteq\; \begin{cases} \emptyset, \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcup\{\mathrm{LV}_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\}, \text{otherwise} \end{cases}
$$

$$
\begin{array}{rcl}
\mathrm{LV}_{entry}(\ell) & \supseteq & (\mathrm{LV}_{exit}(\ell) \backslash \mathit{kill}_{\mathrm{LV}}([B]^{\ell}) \cup \mathit{gen}_{\mathrm{LV}}([B]^{\ell}) \\
 & & \text{where } [B]^{\ell} \in \mathit{blocks}(S_\star)
\end{array}
$$

LV Solutions to $LV^=$ and $LV^\subseteq$

Consider collections $live = (live_{\text{entry}}, live_{\text{exit}})$ of functions:

$$live_{\text{entry}} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$
$$live_{\text{exit}} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

If *live* solves $LV^=$ for a statement S we write:

$$live \models LV^=(S)$$

If *live* solves $LV^\subseteq$ for a statement S we write:

$$live \models LV^\subseteq(S)$$

Theorem 1

Given a label consistent program $S_\star$.

If

- $live \models LV^=(S_\star)$

then

- $live \models LV^\subseteq(S_\star)$.

That is: The least solution of $LV^=(S_\star)$ coincides with the least solution to $LV^\subseteq(S_\star)$.

Theorem 1 - Proof [Not for Exam]

Proof.

If *live* $\models LV^{=}(S_\star)$ also *live* $\models LV^{\subseteq}(S_\star)$ as "$\supseteq$" includes "$=$".

To show that $LV^{=}(S_\star)$ and $LV^{\subseteq}(S_\star)$ have the same least solution consider the iteration operator $\mathbf{F} = \mathbf{F}_{LV} = \mathbf{F}^{S}_{LV}$

$$\begin{array}{lll} \mathit{live} \models LV^{\subseteq}(S_\star) & \text{iff} & \mathit{live} \sqsupseteq \mathbf{F}(\mathit{live}) \\ \mathit{live} \models LV^{=}(S_\star) & \text{iff} & \mathit{live} = \mathbf{F}(\mathit{live}) \end{array}$$

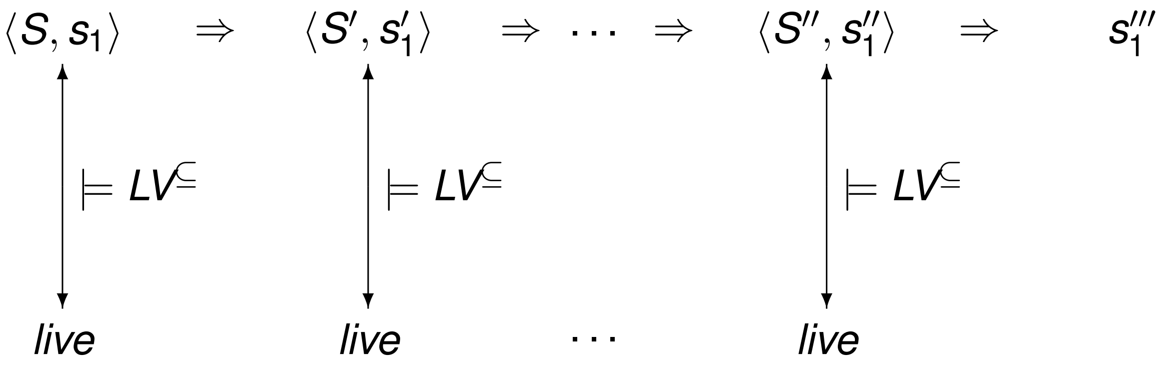
By Tarski's Fixed Point Theorem we have:

$$\mathit{lfp}(\mathbf{F}) = \bigsqcap\{\mathit{live} \mid \mathit{live} \sqsupseteq \mathbf{F}(\mathit{live})\} = \bigsqcap\{\mathit{live} \mid \mathit{live} = \mathbf{F}(\mathit{live})\}.$$

Since $\mathit{lfp}(\mathbf{F}) = \mathbf{F}(\mathit{lfp}(\mathbf{F}))$ and $\mathit{lfp}(\mathbf{F}) \sqsupseteq \mathbf{F}(\mathit{lfp}(\mathbf{F}))$ we see that we get the same least solutions. □

Preservation of Solution

During the (actual) execution of any program $S_\star$ a solution to the Live Variable analysis $LV^{\subseteq}(S_\star)$ remains a solution.



Lemma 2

Given a label consistent program S_1.

If

- $live \models LV^{\subseteq}(S_1)$ and
- $flow(S_1) \supseteq flow(S_2)$ and
- $blocks(S_1) \supseteq blocks(S_2)$

then

- $live \models LV^{\subseteq}(S_2)$

with S_2 being label consistent.

Proof [Not for Exam].

If S_1 is label consistent and $blocks(S_1) \supseteq blocks(S_2)$ then S_2 is also label consistent.

If $live \models LV^{\subseteq}(S_1)$ then $live$ also satisfy each constraint in $LV^{\subseteq}(S_2)$ and hence $live \models LV^{\subseteq}(S_2)$. □

Lemma 3

Given a label consistent program S.

If

- $live \models LV^{\subseteq}(S)$ and
- $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$

then

- $live \models LV^{\subseteq}(S')$.

Proof [Not for Exam].

Follows directly from Lemma 1 and Lemma 2. □

Lemma 4

Given a label consistent program S.

If

- $live \models LV^{\subseteq}(S)$

then for all $(\ell, \ell') \in flow(S)$ we have:

- $live_{\text{exit}}(\ell) \supseteq live_{\text{entry}}(\ell')$

Proof [Not for Exam].
Follows immediately from the construction of $LV^{\subseteq}(S)$. $\square$

Correctness Relation

Assume that V is a set of *live variables*.

Define the correctness relation via

$$s_1 \sim_V s_2 \text{ iff } \forall x \in V : s_1(x) = s_2(x).$$

In other word:
Two states are equivalent iff for all live variables – i.e. all "practical purposes" – the states s_1 and s_2 agree on the variables in V.

Example

Consider $[x := y + z]^\ell$ and $V_1 = \{y, z\}$ and $V_2 = \{x\}$.

$$s_1 \sim_{V_1} s_2 \text{ means } s_1(y) = s_2(y) \ \wedge \ s_1(z) = s_2(z).$$

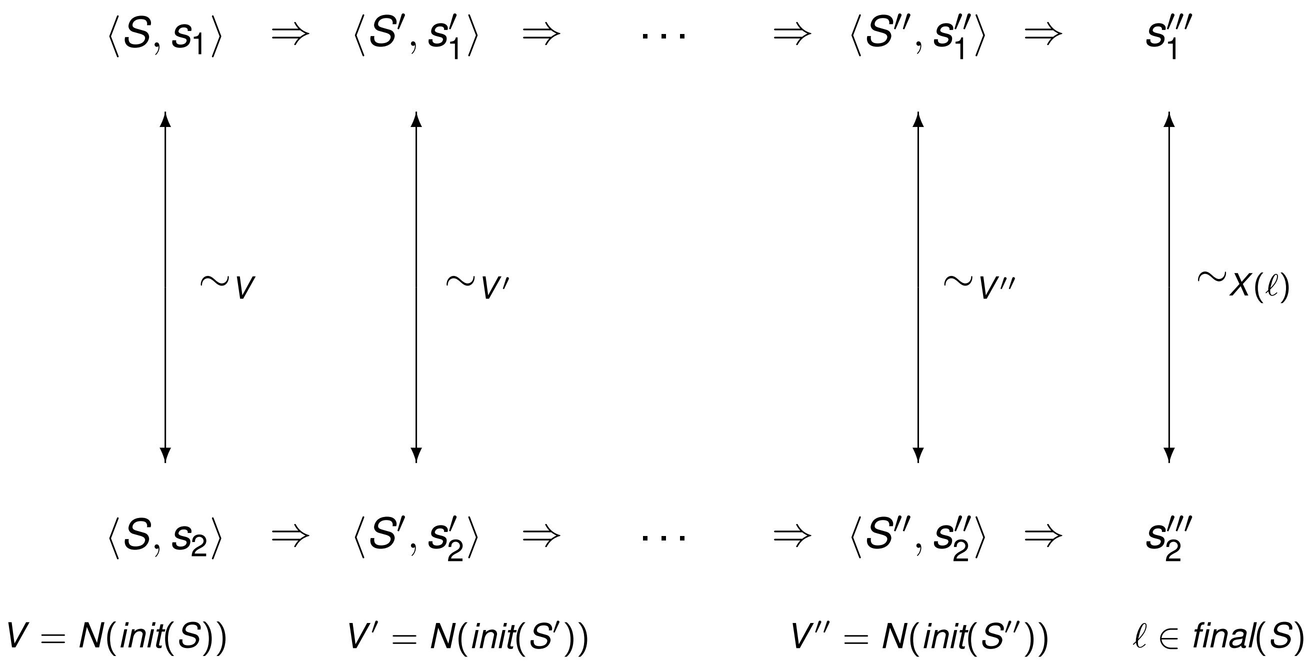
$$s_1 \sim_{V_2} s_2 \text{ means } s_1(x) = s_2(x).$$

Assume $\langle [x := y + z]^\ell, s_1 \rangle \Rightarrow s'_1$, $\langle [x := y + z]^\ell, s_2 \rangle \Rightarrow s'_2$ then

$$s_1 \sim_{V_1} s_2 \text{ ensures } s'_1 \sim_{V_2} s'_2.$$

If $V_2 = \mathrm{LV}_{exit}(\ell)$ thus is the set of live variables after $[x := y + z]^\ell$ then $V_1 = \mathrm{LV}_{entry}(\ell)$ is the set of live variables before $[x := y + z]^\ell$.

Correctness of *LV* Analysis



Short-hand notation: $N(\ell) = live_{\text{entry}}(\ell)$ and $X(\ell) = live_{\text{exit}}(\ell)$.

Lemma 5

Given a label consistent program S.

If

- $live \models LV^{\subseteq}(S)$

then

- $s_1 \sim_{live_{\text{exit}}(\ell)} s_2$ implies $s_1 \sim_{live_{\text{entry}}(\ell')} s_2$ for all $(\ell, \ell') \in flow(S)$.

Proof [Not for Exam].
Follows directly from Lemma 4 and the definition of $\sim_V$. □

Theorem 2

Given a label consistent program S.

If

- $live \models LV^{\subseteq}(S)$

then

(i) If $\langle S, s_1 \rangle \Rightarrow \langle S', s_1' \rangle$ and $s_1 \sim_{live_{\text{entry}}(init(S))} s_2$ then there exists s_2' such that $\langle S, s_2 \rangle \Rightarrow \langle S', s_2' \rangle$ and $s_1' \sim_{live_{\text{entry}}(init(S'))} s_2'$.

(ii) If $\langle S, s_1 \rangle \Rightarrow s_1'$ and $s_1 \sim_{live_{\text{entry}}(init(S))} s_2$ then there exists s_2' such that $\langle S, s_2 \rangle \Rightarrow s_2'$ and $s_1' \sim_{live_{\text{exit}}(init(S))} s_2'$.

Theorem 2 - Proof [Not for Exam]

Proof.

The proof is by induction on the shape of the inference tree.

(ass): ...

(skip): ...

(seq^1): ...

(seq^T): ...

(if^T): ...

(if^F): ...

(wh^T): ...

(wh^F): ...

Theorem 2 - Proof (ass) [Not for Exam]

Proof (cont).

The proof is by induction on the shape of the inference tree.

(ass): We have $\langle [\mathtt{x} := a]^\ell, s_1 \rangle \Rightarrow s_1[x \mapsto [\![a]\!] s_1]$ and from the specification of the constraints:

$$live_{\text{entry}}(\ell) = (live_{\text{exit}}(\ell) \backslash \{x\}) \cup FV(a)$$

and therefore

$$s_1 \sim_{live_{\text{entry}}(\ell)} s_2 \text{ implies } [\![a]\!](s_1) = [\![a]\!](s_2)$$

because the value of a depends only on variables in it.

Thus with $s_2' = s_2[x \mapsto [\![a]\!]_{\mathcal{A}} s_2]$ we have $s_1'(x) = s_2'(x)$ and thus $s_1' \sim_{live_{\text{exit}}(\ell)} s_2'$.

Theorem 2 - Proof (skip) [Not for Exam]

Proof (cont).

(skip): We have $\langle[\texttt{skip}]^\ell, s_1\rangle \Rightarrow s_1$ and from the specification of the constraints we get:

$$live_{\text{entry}}(\ell) = (live_{\text{exit}}(\ell)\backslash\emptyset) \cup \emptyset = live_{\text{exit}}(\ell)$$

Thus taking s_2' to be s_2 we get $s_1' \sim_{live_{\text{exit}}(\ell)} s_2'$ as required.

Theorem 2 - Proof (seq[1]) [Not for Exam]

Proof (cont).

(seq[1]): We have $\langle S_1 ; S_2, s_1\rangle \Rightarrow \langle S_1' ; S_2, s_1'\rangle$ because of $\langle S_1, s_1\rangle \Rightarrow \langle S_1', s_1'\rangle$.

By construction we have $flow(S_1; S_2) \supseteq flow(S_1)$ and also $blocks(S_1; S_2) \supseteq blocks(S_1)$, thus by Lemma 2 $live \models LV^{\subseteq}(S_1)$ and by the induction hypothesis there exists a s_2' such that

$$\langle S_1, s_2\rangle \Rightarrow \langle S_1', s_2'\rangle \text{ and } s_1' \sim_{live_{\text{entry}}(init(S_1'))} s_2'$$

and the result follows.

Theorem 2 - Proof (seq^T) [Not for Exam]

Proof (cont).

(seq^T): We have $\langle S_1; S_2, s_1\rangle \Rightarrow \langle S_2, s_1'\rangle$ because of $\langle S_1, s_1\rangle \Rightarrow s_1'$. Again by Lemma 2, *live* is a solution to $LV^{\subseteq}(S_1)$ and thus by induction hypothesis there exists a s_2' such that

$$\langle S_1, s_2\rangle \Rightarrow s_2' \text{ and } s_1' \sim_{live_{exit}(init(S_1))} s_2'$$

Now we have:

$$\{(\ell, init(S_2)) \mid \ell \in final(S_1)\} \subseteq flow(S_1; S_2)$$

and by Lemma 1, $final(S_1) = \{init(S_1)\}$. Thus by Lemma 5

$$s_1' \sim_{live_{entry}(init(S_2))} s_2'$$

and the result follows.

Theorem 2 - Proof (if^T) & (if^F) [Not for Exam]

Proof (cont).

(if^T): We have $\langle \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2, s_1\rangle \Rightarrow \langle S_1, s_1\rangle$ with $[\![b]\!](s_1) = \textbf{true}$.
Since $s_1 \sim_{live_{entry}(\ell)} s_2$ and $live_{entry}(\ell) \supseteq FV(b)$ we also have $[\![b]\!](s_2) = \textbf{true}$ (the value of b is only dependent on the variables occurring in it) and thus

$$\langle \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2, s_2\rangle \Rightarrow \langle S_1, s_2\rangle$$

From the constraints we get $live_{entry}(\ell) \supseteq live_{exit}(\ell)$ and hence $s_1 \sim_{live_{exit}(\ell)} s_2$.
Since $(\ell, init(S_1)) \in flow(S)$ Lemma 5 gives $s_1 \sim_{live_{entry}(init(S_1))} s_2$ as required.

(if^F): similar to case (if^T).

Theorem 2 - Proof (wh^T) [Not for Exam]

Proof (cont).

(wh^T): $\langle \texttt{while } [b]^\ell \texttt{ do } S, s_1\rangle \Rightarrow \langle S\texttt{; while } [b]^\ell \texttt{ do } S, s_1\rangle$ with $[\![b]\!](s_1) = \textbf{true}$.
Since $s_1 \sim_{live_{\text{entry}}(\ell)} s_2$ and $live_{\text{entry}}(\ell) \supseteq FV(b)$ we also have $[\![b]\!](s_2) = \textbf{true}$ and thus

$$\langle \texttt{while } [b]^\ell \texttt{ do } S, s_2\rangle \Rightarrow \langle S\texttt{; while } [b]^\ell \texttt{ do } S, s_2\rangle$$

Again since $live_{\text{entry}}(\ell) \supseteq live_{\text{exit}}(\ell)$ we have $s_1 \sim_{live_{\text{exit}}(\ell)} s_2$ and then

$$s_1 \sim_{live_{\text{entry}}(init(S))} s_2$$

from Lemma 5 as $(\ell, init(S)) \in flow(\texttt{while } [b]^\ell \texttt{ do } S)$.

Theorem 2 - Proof (wh^F) [Not for Exam]

Proof (cont).

(wh^F): We have $\langle \texttt{while } [b]^\ell \texttt{ do } S, s_1\rangle \Rightarrow s_1$ with $[\![b]\!](s_1) = \textbf{false}$.

Since $s_1 \sim_{live_{\text{entry}}(\ell)} s_2$ and $live_{\text{entry}}(\ell) \supseteq FV(b)$ and we also have $[\![b]\!](s_2) = \textbf{false}$ and thus:

$$\langle \texttt{while } [b]^\ell \texttt{ do } S, s_2\rangle \Rightarrow s_2.$$

From the specification of $LV^{\subseteq}$ we have $live_{\text{entry}}(\ell) \supseteq live_{\text{exit}}(\ell)$ and thus $s_1 \sim_{live_{\text{exit}}(\ell)} s_2$.

□

Corollary 1

Given a label consistent program S.

If

- $live \models LV^{\subseteq}(S)$

then

(i) If $\langle S, s_1 \rangle \Rightarrow^* \langle S', s'_1 \rangle$ and $s_1 \sim_{live_{entry}(init(S))} s_2$ then there exists s'_2 such that $\langle S, s_2 \rangle \Rightarrow^* \langle S', s'_2 \rangle$ and $s'_1 \sim_{live_{entry}(init(S'))} s'_2$.

(ii) If $\langle S, s_1 \rangle \Rightarrow^* s'_1$ and $s_1 \sim_{live_{entry}(init(S))} s_2$ then there exists s'_2 such that $\langle S, s_2 \rangle \Rightarrow^* s'_2$ and $s'_1 \sim_{live_{exit}(\ell)} s'_2$ for some $\ell \in final(S)$.

Proof [Not for Exam].

The proof is by induction on the length of the derivation sequences and uses Theorem 2. □