Formalizing Single-assignment Program Verification: an Adaptation-complete Approach

Cláudio Belo Lourenço Maria João Frade Jorge Sousa Pinto

HASLab/INESC TEC & Universidade do Minho, Portugal

April 6, 2016

Context

Program Verification

- Establishing the **correctness of software** w.r.t. specifications
- Deductive verification achieves this by using **program logics**, relying on user-provided contracts and loop invariants

Trends in modern program verifiers

- Intermediate language tailored for verification
- Single-assignment (SA) form
- Verification condition generator + SMT solver

Gap between program verifiers and theory

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Hoare Logic

- Introduces the notion of **Hoare triple**

$$\{\phi\}\ C\ \{\psi\}$$

- Triples are interpreted based on the **standard semantics** of the programming language

$$\models \{\phi\}\ C\ \{\psi\}$$

- A **proof system** for reasoning about program correctness - system H - sound and (relatively) complete

$$\vdash_{\mathsf{H}} \{\phi\}\ C\ \{\psi\}$$

Adaptation-completeness

If $\models \{n \geq 0 \wedge n_{aux} = n\}\,\text{Fact}\,\{\text{f} = \text{n}_{\text{aux}}!\}$

then $\models \{n = 2\}\,\text{Fact}\,\{\text{f} = 2!\}$

Thus one expects the following derivation to be possible

$$\frac{\begin{array}{c}\{n \geq 0 \wedge n_{aux} = n\}\,\text{Fact}\,\{\text{f} = \text{n}_{\text{aux}}!\}\\ \vdots\end{array}}{\{2 = n\}\,\text{Fact}\,\{\text{f} = 2!\}}$$

If this is always the case the system is called **adaptation-complete**

Adaptation-completeness

If $\models \{n \geq 0 \wedge n_{aux} = n\}\,\mathsf{Fact}\,\{\mathsf{f} = \mathsf{n_{aux}}!\}$

then $\models \{n = 2\}\,\mathsf{Fact}\,\{\mathsf{f} = 2!\}$

Thus one expects the following derivation to be possible

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\,\mathsf{Fact}\,\{\mathsf{f} = \mathsf{n_{aux}}!\} \\ \vdots}{\{2 = n\}\,\mathsf{Fact}\,\{\mathsf{f} = 2!\}}$$

If this is always the case the system is called **adaptation-complete**

Hoare logic is not adaptation-complete

The **consequence** rule of Hoare logic

$$\frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}} \text{ if } \begin{array}{l}\phi' \to \phi \text{ and} \\ \psi \to \psi'\end{array}$$

cannot be applied here:

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\ \text{Fact}\ \{f = n_{aux}!\}}{\{n = 2\}\ \text{Fact}\ \{f = 2!\}} \text{ if } \begin{array}{l} n = 2 \to n \geq 0 \wedge n_{aux} = n \text{ and} \\ f = n_{aux}! \to f = 2! \end{array}$$

In 1998, Kleymann proposed an adaptation-complete inference system for Hoare Logic

$$(\text{conseq}_K) \quad \frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}} \quad \begin{array}{l}\text{if } \forall Z.\forall \sigma.[\![\phi']\!](Z,\sigma) \to \\ \quad \forall \tau.(\forall Z_1.[\![\phi]\!](Z_1,\sigma) \to [\![\psi]\!](Z_1,\tau)) \\ \quad \to [\![\psi']\!](Z,\tau)\end{array}$$

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Hoare logic is not adaptation-complete

The **consequence** rule of Hoare logic

$$\frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}} \quad \text{if} \quad \begin{array}{l} \phi' \to \phi \ \text{ and} \\ \psi \to \psi' \end{array}$$

cannot be applied here:

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\ \text{Fact}\ \{f = n_{aux}!\}}{\{n = 2\}\ \text{Fact}\ \{f = 2!\}} \quad \text{if} \quad \begin{array}{l} n = 2 \to n \geq 0 \wedge n_{aux} = n \quad \text{and} \\ f = n_{aux}! \to f = 2! \end{array}$$

In 1998, Kleymann proposed an adaptation-complete inference system for Hoare Logic

$$(\text{conseq}_K) \quad \frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}} \quad \text{if} \ \forall Z. \forall \sigma. [\![\phi']\!](Z,\sigma) \to \forall \tau. (\forall Z_1. [\![\phi]\!](Z_1,\sigma) \to [\![\psi]\!](Z_1,\tau)) \to [\![\psi']\!](Z,\tau)$$

Hoare logic is not adaptation-complete

The **consequence** rule of Hoare logic

$$\frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}}\ \text{if}\quad \begin{array}{l}\phi' \to \phi\ \ \text{and}\\ \psi \to \psi'\end{array}$$

cannot be applied here:

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\ \text{Fact}\ \{f = n_{aux}!\}}{\{n = 2\}\ \text{Fact}\ \{f = 2!\}}\quad \text{if}\quad \begin{array}{l} n = 2 \to n \geq 0 \wedge n_{aux} = n \quad \text{and} \\ f = n_{aux}! \to f = 2! \end{array}$$

In 1998, Kleymann proposed an adaptation-complete inference system for Hoare Logic

$$(\text{conseq}_K)\qquad \frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}}\quad \begin{array}{l}\text{if}\ \ \forall Z.\forall \sigma.[\![\phi']\!](Z,\sigma) \to \\ \quad \forall \tau.(\forall Z_1.[\![\phi]\!](Z_1,\sigma) \to [\![\psi]\!](Z_1,\tau)) \\ \quad \to [\![\psi']\!](Z,\tau)\end{array}$$

Dijkstra's predicate transformers

Commonly used in the generation of **verification conditions**

The *weakest precondition* of a program w.r.t. a postcondition is given by the function wp, where:

$$
\begin{aligned}
\mathsf{wp}(\mathbf{skip}, \psi) &= \psi \\
\mathsf{wp}(x := e, \psi) &= \psi[e/x] \\
\mathsf{wp}(C_1; C_2, \psi) &= \mathsf{wp}(C_1, \mathsf{wp}(C_2, \psi)) \\
\mathsf{wp}(\mathbf{if}\ b\ \mathbf{then}\ C_t\ \mathbf{else}\ C_f, \psi) &= (b \rightarrow \mathsf{wp}(C_t, \psi)) \wedge (\neg b \rightarrow \mathsf{wp}(C_f, \psi))
\end{aligned}
$$

...

Exponential explosion - example

Let C_n be the program:

```
if (b1) then skip else skip;
...
if (bn) then skip else skip;
```

The generated weakest preconditions are as follows:

1. $\text{wp}(C_1, \psi) = (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)$
2. $\text{wp}(C_2, \psi) = (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi))$
3. $\text{wp}(C_3, \psi) =$
   $(b_3 \rightarrow (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)))$
   $\wedge (\neg b_3 \rightarrow (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)))$
4. ...

Exponential explosion - example

Let C_n be the program:

if (b_1) **then skip else skip** ;
...
if (b_n) **then skip else skip** ;

The generated weakest preconditions are as follows:

1. $\mathsf{wp}(C_1, \psi) = (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)$
2. $\mathsf{wp}(C_2, \psi) = (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi))$
3. $\mathsf{wp}(C_3, \psi) =$
   $(b_3 \rightarrow (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)))$
   $\wedge (\neg b_3 \rightarrow (b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)) \wedge (\neg b_2 \rightarrow (b_1 \rightarrow \psi) \wedge (\neg b_1 \rightarrow \psi)))$
4. ...

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Exponential explosion - example

Let C_n be the program:

```
if (b1) then skip else skip;
...
if (bn) then skip else skip;
```

The generated weakest preconditions are as follows:

1. $\mathrm{wp}(C_1, \psi) = (b_1 \to \psi) \wedge (\neg b_1 \to \psi)$
2. $\mathrm{wp}(C_2, \psi) = (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi))$
3. $\mathrm{wp}(C_3, \psi) =$
   $(b_3 \to (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)))$
   $\wedge (\neg b_3 \to (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)))$
4. ...

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Exponential explosion - example

Let C_n be the program:

```
if (b1) then skip else skip;
...
if (bn) then skip else skip;
```

The generated weakest preconditions are as follows:

1. $\mathsf{wp}(C_1, \psi) = (b_1 \to \psi) \wedge (\neg b_1 \to \psi)$
2. $\mathsf{wp}(C_2, \psi) = (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi))$
3. $\mathsf{wp}(C_3, \psi) =$
   $(b_3 \to (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)))$
   $\wedge\ (\neg b_3 \to (b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)) \wedge (\neg b_2 \to (b_1 \to \psi) \wedge (\neg b_1 \to \psi)))$
4. ...

Single-assignment programs

The exponential explosion can be avoided if programs are converted first into **passive-form** (similar to single-assignment programs) [Flanagan & Saxe, 2001]

Let C^{SA} be the program

```
if (x0 < 0) then
  x1 := -x0
else
  x1 := x0
```

Calculating the WP is now direct

$$wp^*(C^{SA}, x_1 > 0) = ((x_0 < 0 \land x_1 = -x_0) \lor (\neg(x_0 < 0) \land x_1 = x_0)) \rightarrow x_1 > 0$$

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Single-assignment programs

The exponential explosion can be avoided if programs are converted first into **passive-form** (similar to single-assignment programs) [Flanagan & Saxe, 2001]

Let C^{SA} be the program

$$\begin{array}{l} \textbf{if}\ (x_0 < 0)\ \textbf{then} \\ \quad x_1 := -x_0 \\ \textbf{else} \\ \quad x_1 := x_0 \end{array}$$

Calculating the WP is now direct

$$\begin{array}{rcl} \mathrm{wp}^*(C^{SA}, x_1 > 0) & = & ((x_0 < 0 \land x_1 = -x_0) \lor (\neg(x_0 < 0) \land x_1 = x_0)) \\ & & \rightarrow \boldsymbol{x_1 > 0} \end{array}$$

Contributions

We **formalize and prove** a verification technique based on the translation of programs into a single-assignment intermediate form:

- a novel notion of **annotated SA programs**
- a **translation** of While programs (resp. Hoare triples) into SA programs (resp. SA Hoare triples)
- a **logic** and an **efficient VCGen** to reason about SA programs
- an **adaptation-complete** extension of the logic

helping to bridge the gap between program verifiers and theoretical foundations

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Setting

While programs

$$\textbf{Comm} \ni C ::= \textbf{skip} \mid x := e \mid C\,;\,C \mid \textbf{if}\ b\ \textbf{then}\ C\ \textbf{else}\ C$$
$$\mid \textbf{while}\ b\ \textbf{do}\ C$$

Annotated while programs

$$\textbf{AComm} \ni C ::= \textbf{skip} \mid x := e \mid C\,;\,C \mid \textbf{if}\ b\ \textbf{then}\ C\ \textbf{else}\ C$$
$$\mid \textbf{while}\ b\ \textbf{do}\ \{\theta\}\ C$$

Erasing annotations

$\lfloor . \rfloor : \textbf{AComm} \to \textbf{Comm}$

Setting

While programs

$$\mathbf{Comm} \ni C \ ::= \ \mathbf{skip} \mid x := e \mid C\,;\,C \mid \mathbf{if}\ b\ \mathbf{then}\ C\ \mathbf{else}\ C$$
$$\mid \mathbf{while}\ b\ \mathbf{do}\ C$$

Annotated while programs

$$\mathbf{AComm} \ni C \ ::= \ \mathbf{skip} \mid x := e \mid C\,;\,C \mid \mathbf{if}\ b\ \mathbf{then}\ C\ \mathbf{else}\ C$$
$$\mid \mathbf{while}\ b\ \mathbf{do}\,\{\theta\}\ C$$

Erasing annotations

$\lfloor . \rfloor : \mathbf{AComm} \to \mathbf{Comm}$

Goal directed logic - system Hg

$$(\text{skip})\ \frac{}{\{\phi\}\ \textbf{skip}\ \{\psi\}}\ \text{if } \phi \to \psi \qquad (\text{assign})\ \frac{}{\{\phi\}\ x := e\ \{\psi\}}\ \text{if } \phi \to \psi[e/x]$$

$$(\text{seq})\ \frac{\{\phi\}\ C_1\ \{\theta\} \qquad \{\theta\}\ C_2\ \{\psi\}}{\{\phi\}\ C_1\ ;\ C_2\ \{\psi\}} \qquad (\text{if})\ \frac{\{\phi \land b\}\ C_t\ \{\psi\} \qquad \{\phi \land \neg b\}\ C_f\ \{\psi\}}{\{\phi\}\ \textbf{if}\ b\ \textbf{then}\ C_t\ \textbf{else}\ C_f\ \{\psi\}}$$

$$(\text{while})\ \frac{\{\theta \land b\}\ C\ \{\theta\}}{\{\phi\}\ \textbf{while}\ b\ \textbf{do}\ \{\theta\}\ C\ \{\psi\}}\ \text{if}\ \begin{array}{l} \phi \to \theta \ \text{ and} \\ \theta \land \neg b \to \psi \end{array}$$

- Hg is shown to be **sound** w.r.t. system H
- A program C is said to be **correctly annotated** w.r.t. (ϕ, ψ), if $\vdash_{\mathsf{H}} \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$ implies $\vdash_{\mathsf{Hg}} \{\phi\}\ C\ \{\psi\}$

Factorial example

$\{n \geq 0 \land n_{aux} = n\}$

$f := 1\,;$

$i := 1\,;$

while $i \leq n$ **do** $\{f = (i-1)! \land i \leq n+1 \land n_{aux} = n\}$

$\{$

$\quad f := f * i\,;$

$\quad i := i + 1$

$\}$

$\{f = n_{aux}!\}$

Factorial example

$f := 1\,;$

$i := 1\,;$

$\textbf{while } i \leq n \textbf{ do} \, \{f = (i-1)! \wedge i \leq n+1 \wedge n_{aux} = n\}$

$\{$

$\quad f := f * i\,;$

$\quad i := i + 1$

$\}$

Factorial example

$f_1 := 1\,;$

$i_1 := 1\,;$

$\mathcal{I}$

while $(i_2 \leq n_0)$ **do** $\{f_2 = (i_2 - 1)! \wedge i_2 \leq n_0 + 1 \wedge n_{aux_0} = n_0\}$

$\{$

$\quad f_3 := f_2 * i_2\,;$

$\quad i_3 := i_2 + 1$

$\quad \mathcal{U}$

$\}$

Iterating single-assignment language

$$\mathbf{AComm}^{\mathsf{sa}} \ni C \ ::= \mathbf{skip} \mid x := e \mid C\,;\,C \mid \mathbf{if}\ b\ \mathbf{then}\ C\ \mathbf{else}\ C$$
$$\mid \mathbf{for}\ (\mathcal{I}, b, \mathcal{U})\ \mathbf{do}\ \{\theta\}\ C$$

Restrictions on the use of variables imposed

- $x := e \in \mathbf{AComm}^{\mathsf{sa}}$ only if $x \notin \mathsf{Vars}(e)$
- $C_1\,;\,C_2 \in \mathbf{AComm}^{\mathsf{sa}}$ only if $C_1, C_2 \in \mathbf{AComm}^{\mathsf{sa}}$ and $\mathsf{Vars}(C_1) \cap \mathsf{Asgn}(C_2) = \emptyset$
- ...

$\mathcal{W} : \mathbf{AComm}^{\mathsf{sa}} \rightarrow \mathbf{AComm}$

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Factorial example - single-assignment

$$
\begin{array}{l}
f_1 := 1\,; \\
i_1 := 1\,; \\
\textbf{for}\ (\mathcal{I}, i_2 \leq n_0,\ \mathcal{U})\ \textbf{do}\,\{ \\
f_2 = (i_2 - 1)! \land i_2 \leq n_0 + 1\ \land\ n_{aux_0} = n_0\} \\
\{ \\
\quad f_3 := f_2 * i_2\,; \\
\quad i_3 := i_2 + 1 \\
\}
\end{array}
$$

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Factorial example - single-assignment

$f_1 := 1\,;$

$i_1 := 1\,;$

$\textbf{for}\ (\{i_2 := i_1\ ;\ f_2 := f_1\}, i_2 \leq n_0, \{i_2 := i_3\ ;\ f_2 := f_3\})\ \textbf{do}\ \{$

$f_2 = (i_2 - 1)! \wedge i_2 \leq n_0 + 1\ \wedge\ n_{aux_0} = n_0\}$

$\{$

$\quad f_3 := f_2 * i_2\,;$

$\quad i_3 := i_2 + 1$

$\}$

HASLab HIGH-ASSURANCE SOFTWARE LABORATORY

Factorial example - single-assignment

$\{n_0 \geq 0 \wedge n_{aux_0} = n_0\}$

$f_1 := 1\,;$

$i_1 := 1\,;$

$\textbf{for}\ (\{i_2 := i_1\,;\ f_2 := f_1\}, i_2 \leq n_0, \{i_2 := i_3\,;\ f_2 := f_3\})\ \textbf{do}\,\{$

$f_2 = (i_2 - 1)! \wedge i_2 \leq n_0 + 1\}$

$\{$

$\quad f_3 := f_2 * i_2\,;$

$\quad i_3 := i_2 + 1$

$\}$

$\{f_2 = n_{aux_0}!\}$

SA translation

- We let $\phi \# C$ denote $\mathsf{Asgn}(C) \cap \mathsf{FV}(\phi) = \emptyset$
- We call $\{\phi\}\, C\, \{\psi\}$ an **SA triple** if $C \in \mathbf{AComm}^{\mathsf{sa}}$ and $\phi \# C$
- A function

  $\mathcal{T} : \mathbf{Assert} \times \mathbf{AComm} \times \mathbf{Assert} \to \mathbf{Assert} \times \mathbf{AComm}^{\mathsf{sa}} \times \mathbf{Assert}$

  is said to be a **single-assignment translation** if when $\mathcal{T}(\phi, C, \psi) = (\phi', C', \psi')$ we have $\phi' \# C'$, and:

  1. If $\models \{\phi'\}\, \lfloor \mathcal{W}(C') \rfloor\, \{\psi'\}$, then $\models \{\phi\}\, \lfloor C \rfloor\, \{\psi\}$
  2. If $\vdash_{\mathsf{Hg}} \{\phi\}\, C\, \{\psi\}$, then $\vdash_{\mathsf{Hg}} \{\phi'\}\, \mathcal{W}(C')\, \{\psi'\}$

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Inference system for annotated SA programs - system Hsa

$$\text{(skip)} \quad \overline{\{\phi\}\,\textbf{skip}\,\{\phi \wedge \top\}} \qquad \text{(assign)} \quad \overline{\{\phi\}\, x := e\,\{\phi \wedge x = e\}}$$

$$\text{(seq)} \quad \frac{\{\phi\}\, C_1\,\{\phi \wedge \psi_1\} \qquad \{\phi \wedge \psi_1\}\, C_2\,\{\phi \wedge \psi_1 \wedge \psi_2\}}{\{\phi\}\, C_1\,;\, C_2\,\{\phi \wedge \psi_1 \wedge \psi_2\}}$$

$$\text{(if)} \quad \frac{\{\phi \wedge b\}\, C_t\,\{\phi \wedge b \wedge \psi_t\} \qquad \{\phi \wedge \neg b\}\, C_f\,\{\phi \wedge \neg b \wedge \psi_f\}}{\{\phi\}\,\textbf{if}\ b\ \textbf{then}\ C_t\ \textbf{else}\ C_f\,\{\phi \wedge ((b \wedge \psi_t) \vee (\neg b \wedge \psi_f))\}}$$

$$\text{(for)} \quad \frac{\{\theta \wedge b\}\, C\,\{\theta \wedge b \wedge \psi\}}{\{\phi\}\,\textbf{for}\ (\mathcal{I}, b, \mathcal{U})\ \textbf{do}\,\{\theta\}\, C\,\{\phi \wedge \theta \wedge \neg b\}} \quad \text{if} \quad \begin{array}{l} \phi \rightarrow \mathcal{I}(\theta) \ \ \text{and} \\ \theta \wedge b \wedge \psi \rightarrow \mathcal{U}(\theta) \end{array}$$

- Hsa is shown to be **sound** w.r.t H for SA triples
- Hsa is shown to be **complete** w.r.t. Hg for SA triples

Adaptation-complete system - system Hsa$^+$

Let Hsa$^+$ be the system Hsa with the addition of the following rule

$$\frac{\{\phi\}\ C\ \{\phi \wedge \psi\}}{\{\phi'\}\ C\ \{\phi' \wedge (\forall\, \vec{x}.\, \phi \rightarrow \psi)\}} \quad \text{if} \quad \phi \# C$$

$$\vec{x} = \mathsf{FV}(\phi) \setminus (\mathsf{FV}(\phi') \cup \mathsf{Vars}(C))$$

Hsa$^+$ is an **adaptation-complete** system for SA programs

In this system the following derivation is possible

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\ \mathsf{Fact}^{\mathsf{sa}}\ \{n \geq 0 \wedge n_{aux} = n \wedge f_2 = n_{aux}!\}}{\{n = 2\}\ \mathsf{Fact}^{\mathsf{sa}}\ \{n = 2 \wedge (\forall\, n_{aux}.\, n \geq 0 \wedge n_{aux} = n \rightarrow f_2 = n_{aux}!)\}}$$

and $\models n = 2 \wedge (\forall\, n_{aux}.\, n \geq 0 \wedge n_{aux} = n \rightarrow f_2 = n_{aux}!) \rightarrow f = 2!$

Adaptation-complete system - system Hsa$^+$

Let Hsa$^+$ be the system Hsa with the addition of the following rule

$$\frac{\{\phi\}\, C\, \{\phi \wedge \psi\}}{\{\phi'\}\, C\, \{\phi' \wedge (\forall\, \vec{x}.\, \phi \rightarrow \psi)\}} \quad \text{if} \quad \phi \# C \quad \vec{x} = \mathsf{FV}(\phi) \setminus (\mathsf{FV}(\phi') \cup \mathsf{Vars}(C))$$

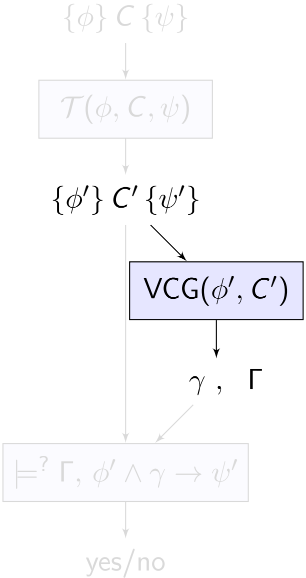
Hsa$^+$ is an **adaptation-complete** system for SA programs

In this system the following derivation is possible

$$\frac{\{n \geq 0 \wedge n_{aux} = n\}\, \mathsf{Fact}^{\mathsf{sa}}\, \{n \geq 0 \wedge n_{aux} = n \wedge f_2 = n_{aux}!\}}{\{n = 2\}\, \mathsf{Fact}^{\mathsf{sa}}\, \{n = 2 \wedge (\forall\, n_{aux}.\, n \geq 0 \wedge n_{aux} = n \rightarrow f_2 = n_{aux}!)\}}$$

and $\models n = 2 \wedge (\forall\, n_{aux}.\, n \geq 0 \wedge n_{aux} = n \rightarrow f_2 = n_{aux}!) \rightarrow f = 2!$

Verification technique



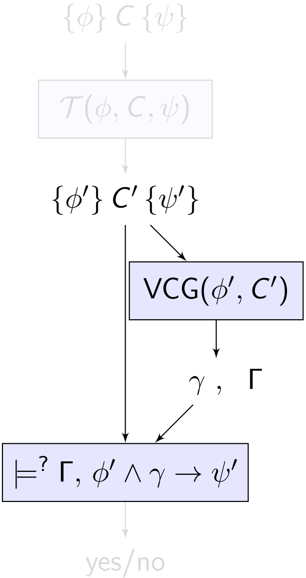
Soundness

If $\models \Gamma, \phi' \wedge \gamma \rightarrow \psi'$ then $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$

Completeness

If $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$ and C is correctly-annotated w.r.t. (ϕ, ψ), then $\models \Gamma, \phi' \wedge \gamma \rightarrow \psi'$

Verification technique



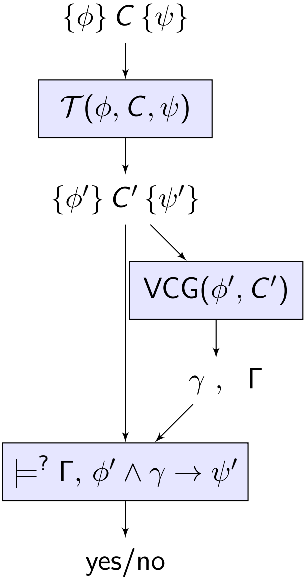
Soundness

If $\models \Gamma, \phi' \wedge \gamma \rightarrow \psi'$ then $\models \{\phi\} \lfloor C \rfloor \{\psi\}$

Completeness

If $\models \{\phi\} \lfloor C \rfloor \{\psi\}$ and C is correctly-annotated w.r.t. (ϕ, ψ), then $\models \Gamma, \phi' \wedge \gamma \rightarrow \psi'$

Verification technique



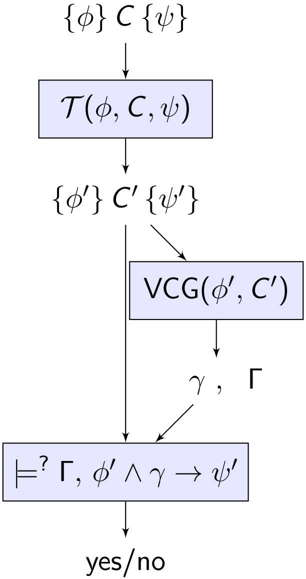
Soundness

If $\models \Gamma, \phi' \wedge \gamma \to \psi'$ then $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$

Completeness

If $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$ and C is correctly-annotated w.r.t. (ϕ, ψ), then $\models \Gamma, \phi' \wedge \gamma \to \psi'$

Verification technique



Soundness

If $\models \Gamma, \phi' \wedge \gamma \to \psi'$ then $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$

Completeness

If $\models \{\phi\}\ \lfloor C \rfloor\ \{\psi\}$ and C is correctly-annotated w.r.t. (ϕ, ψ), then $\models \Gamma, \phi' \wedge \gamma \to \psi'$

Conclusion

- Our work proposes a **theoretical foundation** for program verifiers based on intermediate **single-assignment** form
- Hsa logic for SA programs with annotated loops
  - proved **sound** and **complete**
  - admits **adaptation-complete** extension
  - allows for the generation of **linear-sized VCs**
- As future work we intend to use this formulation to reason about **bounded verification of programs**

Formalizing Single-assignment Program Verification: an Adaptation-complete Approach

Cláudio Belo Lourenço   Maria João Frade   Jorge Sousa Pinto

HASLab/INESC TEC & Universidade do Minho, Portugal

April 6, 2016

Hoare calculus - system H

$$(\text{skip}) \quad \overline{\{\phi\}\ \textbf{skip}\ \{\phi\}} \qquad (\text{assign}) \quad \overline{\{\psi[e/x]\}\ x := e\ \{\psi\}}$$

$$(\text{seq}) \quad \frac{\{\phi\}\ C_1\ \{\theta\} \qquad \{\theta\}\ C_2\ \{\psi\}}{\{\phi\}\ C_1\ ;\ C_2\ \{\psi\}} \qquad (\text{if}) \quad \frac{\{\phi \wedge b\}\ C_t\ \{\psi\} \qquad \{\phi \wedge \neg b\}\ C_f\ \{\psi\}}{\{\phi\}\ \textbf{if}\ b\ \textbf{then}\ C_t\ \textbf{else}\ C_f\ \{\psi\}}$$

$$(\text{while}) \quad \frac{\{\theta \wedge b\}\ C\ \{\theta\}}{\{\theta\}\ \textbf{while}\ b\ \textbf{do}\ C\ \{\theta \wedge \neg b\}} \qquad (\text{conseq}) \quad \frac{\{\phi\}\ C\ \{\psi\}}{\{\phi'\}\ C\ \{\psi'\}} \ \text{ if } \ \begin{array}{l} \phi' \rightarrow \phi \ \text{ and} \\ \psi \rightarrow \psi' \end{array}$$

H is shown to be **sound** and **complete** (in the sense of Cook) w.r.t the semantics of Hoare triples