Some notions of information flow

Jérôme Feret
Laboratoire d'Informatique de l'École Normale Supérieure
INRIA, ÉNS, CNRS
http://www.di.ens.fr/~feret

Friday, the 20th of November, 2020
Syntax

Let $\mathcal{V} \doteq \{V, V_1, V_2, \ldots\}$ be a finite set of variables.
Let $\mathcal{Z} \doteq \{z, \ldots\}$ be the set of relative numbers.
Expressions are polynomial of variables $\mathcal{V}$.

$$E ::= z \mid V \mid E + E \mid E \times E$$

Programs are given by the following grammar:

$$P ::= \text{skip} \mid P;P \mid V := E \mid \text{if } (V \geq 0) \{P\} \text{ else } \{P\} \mid \text{while } (V \geq 0) \{P\}$$
Semantics

We define the semantics \([P] \in \mathcal{F}(\mathcal{V} \to \mathbb{Z}) \cup \Omega)\) of a program \(P\):

- \([\text{skip}] (\rho) = \rho\),
- \([P_1; P_2] (\rho) = \begin{cases} \Omega & \text{if } [P_1] (\rho) = \Omega \\ [P_2] ([P_1] (\rho)) & \text{otherwise} \end{cases}\)
- \([V := E](\rho) = \begin{cases} \Omega & \text{if } \rho = \Omega \\ \rho [V \mapsto \overline{\rho}(E)] & \text{otherwise} \end{cases}\)
- \([\text{if } (V \geq 0) \{P_1\} \text{ else } \{P_2\}] (\rho) = \begin{cases} \Omega & \text{if } \rho = \Omega \\ [P_1] (\rho) & \text{if } \rho(V) \geq 0 \\ [P_2] (\rho) & \text{otherwise} \end{cases}\)
- \([\text{while } (V \geq 0) \{P\}] (\rho) = \begin{cases} \Omega & \text{if } \rho = \Omega \\ \rho' & \text{if } \{\rho'\} = \{\rho' \in \text{Inv} \mid \rho'(V) < 0\} \\ \Omega & \text{otherwise} \end{cases}\)

where \(\text{Inv} = \text{lfp}(X \mapsto \{\rho\} \cup \{\rho'' \mid \exists \rho' \in X, \rho'(V) \geq 0 \text{ and } \rho'' \in [P] (\rho')\})\).
Flow of information

Given a program $P$, we say that the variable $V_1$ flows into the variable $V_2$ if, and only if, the final value of $V_2$ depends on the initial value of $V_1$, which is written $V_1 \Rightarrow_P V_2$.

More formally, $V_1 \Rightarrow_P V_2$ if and only if there exists $\rho \in \mathcal{V} \rightarrow \mathbb{Z}$, $z, z' \in \mathbb{Z}$ such that one of the following three assertions is satisfied:

1. $[P](\rho[V_1 \mapsto z]) \neq \Omega$, $[P](\rho[V_1 \mapsto z']) \neq \Omega$, and $[P](\rho[V_1 \mapsto z])(V_2) \neq [P](\rho[V_1 \mapsto z'])(V_2)$;
2. $[P](\rho[V_1 \mapsto z]) = \Omega$ and $[P](\rho[V_1 \mapsto z']) \neq \Omega$;
3. $[P](\rho[V_1 \mapsto z]) \neq \Omega$ and $[P](\rho[V_1 \mapsto z']) = \Omega$. 
Let $P$ be a program.

We define the following binary relation $\rightarrow_P$ among variables in $\mathcal{V}$:
$V_1 \rightarrow_P V_2$ if and only if there is an assignment in $P$ of the form $V_2 := E$ such that $V_1$ occurs in $E$.

Does $V_1 \Rightarrow_P V_2$ imply that $V_1 \rightarrow_P^* V_2$?
Counter-example

We consider the following program $P$:

$$
P ::= \begin{cases} 
\text{if } (V_1 \geq 0) \\
\{ V_2 := 0 \} \\
\text{else} \\
\{ V_2 := 1 \}
\end{cases}
$$

For any $\rho \in \mathcal{V} \to \mathbb{Z}$, we have $\llbracket P \rrbracket(\rho[V_1 \mapsto 0])(V_2) = 0$; but, $\llbracket P \rrbracket(\rho[V_1 \mapsto 1])(V_2) = 1$; so $V_1 \Rightarrow_P V_2$; But $V_1 \nRightarrow_P^* V_2$. 
For each program point \( p \) in \( P \), we denote by \( test(p) \) the set of variables which occur in the guards of tests and while loops the scope of which contains the program point \( p \).

We define the following binary relation \( \rightarrow \) among variables in \( \mathcal{V} \): \( V_1 \rightarrow_p V_2 \) if and only if there is an assignment in \( P \) of the form \( V_2 := E \) at program point \( p \) such that:

1. either \( V_1 \) occurs in \( E \);
2. or \( V_1 \in test(p) \).

Does \( V_1 \Rightarrow_p V_2 \) imply that \( V_1 \rightarrow^*_P V_2 \)?
Counter-example

We consider the following program $P$:

$$P ::= \text{while } (V_1 \geq 0) \{\text{skip}\}$$

For any $\rho \in V \rightarrow \mathbb{Z}$, we have $[P](\rho[V_1 \mapsto -1]) \neq \Omega$;
but, $[P](\rho[V_1 \mapsto 0]) = \Omega$;
so $V_1 \Rightarrow_P V_2$;
But $V_1 \not\Rightarrow^*_P V_2$. 
Approximation of the information flow

So as to get a sound approximation of the information flow, we have to consider that a variable that is tested in the guard of a loop may flow in any variable.

We define the following binary relation $\to_P$ among variables in $\mathcal{V}$: $V_1 \to V_2$ if and only if there is an assignment in $P$ of the form $V_2 := E$ at program point $p$ such that:

1. either $V_1$ occurs in $E$;
2. or $V_1$ is tested in the guard of a loop;
3. or $V_1 \in test(p)$.

**Theorem 1** If $V_1 \Rightarrow_P V_2$, then $V_1 \to_P^* V_2$!
Limitations

The approximation is highly syntax-oriented.

- It is context-insensitive;
- It is very rough in the case of while loop, $\iff$ we could show statically that some loops always terminate to avoid fictitious dependencies;
- we could detect some invariants to avoid fictitious dependencies.

Other forms of attacks could be modeled in the semantics: an attacker could observe:

- computation time;
- memory assumption;
- heating.

(attacks cannot be exhaustively specified).