

MPRI 2.19 Biochemical Programming

Rule-based Modeling

Model reduction

Some notions of information flow

Jérôme Feret

Laboratoire d'Informatique de l'École Normale Supérieure

INRIA, ÉNS, CNRS

<http://www.di.ens.fr/~feret>

Monday, the 13th of January, 2025

Syntax

Let $\mathcal{V} \triangleq \{V, V_1, V_2, \dots\}$ be a finite set of variables.

Let $\mathbb{Z} \triangleq \{z, \dots\}$ 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:

$$\begin{array}{l} P ::= \text{skip} \\ \quad \mid P; P \\ \quad \mid V := E \\ \quad \mid \text{if } (V \geq 0) \{P\} \text{ else } \{P\} \\ \quad \mid \text{while } (V \geq 0) \{P\} \end{array}$$

Semantics

We define the semantics $\llbracket P \rrbracket \in \mathcal{F}((\mathcal{V} \rightarrow \mathbb{Z}) \cup \Omega)$ of a program P :

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

where $\text{Inv} = \text{lfp } (X \mapsto \{\rho\} \cup \{\rho'' \mid \exists \rho' \in X, \rho'(V) \geq 0 \text{ and } \rho'' \in \llbracket P \rrbracket(\rho')\})$.

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. $\llbracket P \rrbracket(\rho[V_1 \mapsto z]) \neq \Omega$, $\llbracket P \rrbracket(\rho[V_1 \mapsto z']) \neq \Omega$,
and $\llbracket P \rrbracket(\rho[V_1 \mapsto z])(V_2) \neq \llbracket P \rrbracket(\rho[V_1 \mapsto z'])(V_2)$;
2. $\llbracket P \rrbracket(\rho[V_1 \mapsto z]) = \Omega$ and $\llbracket P \rrbracket(\rho[V_1 \mapsto z']) \neq \Omega$;
3. $\llbracket P \rrbracket(\rho[V_1 \mapsto z]) \neq \Omega$ and $\llbracket P \rrbracket(\rho[V_1 \mapsto z']) = \Omega$.

Syntactic approximation (tentative)

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 assignement 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 :

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

For any $\rho \in \mathcal{V} \rightarrow \mathbb{Z}$,
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 $V_1 \not\rightarrow_P^* V_2$.

Syntactic approximation (tentative)

For each program point p in P ,
we denote by $\text{test}(p)$ 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 $V_1 \in \text{test}(p)$.

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 $\rho \in \mathcal{V} \rightarrow \mathbb{Z}$,
we have $\llbracket P \rrbracket(\rho[V_1 \mapsto -1]) \neq \Omega$;
but, $\llbracket P \rrbracket(\rho[V_1 \mapsto 0]) = \Omega$;
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 \rightarrow_P among variables in \mathcal{V} :

$V_1 \rightarrow V_2$ 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 $V_1 \in \text{test}(p)$.

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

Limitations

The approximation is highly syntax-oriented.

- It is context-insensitive;
- It is very rough in the case of while loop,
 \implies 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).