Cours MALG & MOVEX

MALG
Analyse des programmes

Dominique Méry
Telecom Nancy, Université de Lorraine
(22 avril 2025 at 7:54 A.M.)

Année universitaire 2024-2025

Outline

Main concepts

- **Syntax of programs** ($P \in PL$) defines the class of programs for applying the analysis@
- **Semantics** ($[\![P]\!]$) for each program ($P \in PL$)
-

Standard, Collecting and Abstract Semantics

- Abstract interpretation of programs is an approximation of programs semantics
- Correctness proof of the abstract interpretation requires the existence of the standard semantics describing the possible behaviours of programs during their execution.
- The class of properties of program executions is defined by a collecting semantics or static semantics.
- The collecting semantics can be an instrumented version of the standard semantics to gather information about programs executions.
- or the standard semantics reduced to essentials in order to ignore irrelevant details about program execution.
- The collecting semantics provides a sound and relatively complete proof method for the considered class of properties.
- It can be used subsequently as a reference semantics for proving the correctness of all other approximate semantics for that class of properties.
- The abstract semantics usually considers effectively computable properties of programs.
- The soundness of this abstract semantics is proved with respect to the collecting semantics.

Collecting semantics

Examples

- Computation Traces of Program
- Transitive Closure of the program transition relation
- Set of states

The collecting semantics is the semantics which is interesting our analysis and we will consider as collecting semantics the set of states.

Summary of the technique

- *Collecting semantics*
  - Static analysis of a program states a property of program executions defined by a *standard* semantics.
  - Defining a so-called *collecting* semantics defining the strongest static property of interest
  - Collecting semantics defines the class of static analysis, which approximates it
  - State properties are subsets of $\mathbb{I}\times\mathbb{I}\times\mathbb{I}\times\mathbb{I}$ and abstract interpretation executes programs on thse properties
- Approximation
  - Spaces of values should be restricted to computable entities
  - Over-approximation of concrete properties

Small Programming Language

$$
\begin{array}{llll}
Expr & ::= & v & v \in \mathbb{Z} \\
 & | & ? & \\
 & | & x & x \in \mathbb{V} \\
 & | & Expr\ op\ Expr & op \in \{+, -, \times, /\} \\
 & & & \\
cond & ::= & Expr\ relop\ Expr & relop \in \{<, \leq, >, \geq, =, \neq\} \\
 & | & \mathbf{not}\ cond & \\
 & | & cond\ \mathbf{and}\ cond & \\
 & & & \\
stmt & ::= & \ell[x := Expr] & \ell \in \mathbb{C} \\
 & | & \ell[skip] & \\
 & | & \mathbf{if}\ \ell[cond]\ \mathbf{then}\ stmt\ \mathbf{else}\ stmt\ \mathbf{end\ if} & \\
 & | & \mathbf{while}\ \ell[cond]\ \mathbf{do}\ \ stmt\ \mathbf{end\ do} & \\
 & | & stmt; stmt &
\end{array}
$$

Two examples of annotated programs

$\ell_0[X := 0];$
$\ell_1[Y := Y{+}X];$
$\ell_2[skip]$
$\ell_3[X := Y];$

$\ell_0[Q := 0];$
$\ell_1[R := X];$
IF $\ell_5[Y > 0]$
    WHILE $\ell_2[R \geq Y]$
        $\ell_3[Q := Q{+}1];$
        $\ell_4[R := R{-}Y]$
    ENDWHILE
ELSE
    $\ell_6[skip]$
ENDIF

Defining the semantics of the small programming language

- Semantic Domains

$$Mem \stackrel{def}{=} \mathbb{V} \longrightarrow \mathbb{Z}$$
$$States \stackrel{def}{=} \mathbb{C} \times Mem$$

- Semantics for Expressions

$\mathcal{E}[\![v]\!](m) \in \mathcal{P}(\mathbb{Z})$, $e \in Expr, m \in Mem$, $x \in \mathbb{V}$, $op \in \{+, -, \times, /\}$

$$\mathcal{E}[\![v]\!](m) \stackrel{def}{=} \{v\}$$
$$\mathcal{E}[\![?]\!](m) \stackrel{def}{=} \mathbb{Z}$$
$$\mathcal{E}[\![x]\!](m) \stackrel{def}{=} \{m(x)\}$$
$$\mathcal{E}[\![e_1 \ op \ e_2]\!](m) \stackrel{def}{=} \{v | \exists ve_1, ve_2. \left(\begin{array}{l} ve_1 \in \mathcal{E}[\![e_1]\!](m) \\ ve_2 \in \mathcal{E}[\![e_2]\!](m) \\ v = ve_1 \ o \ ve_2 \end{array}\right)\}$$

- Semantics for conditions
  $\mathcal{C}[\![cond]\!](m) \in \mathcal{P}(\mathbb{B})$, $cond \in Cond$,$m \in Mem$, $x \in \mathbb{V}$, $op \in \{+, -, \times, /\}$

$$
\begin{array}{lcl}
tt \in \mathcal{C}[\![e_1 \ relop \ e_2]\!](m) & \stackrel{def}{=} & \exists v_1, v2. \left(\begin{array}{l} v_1 \in \mathcal{E}[\![e_1]\!](m) \\ v_2 \in \mathcal{E}[\![\ e_2]\!](m) \\ v_1 \ relop \ v2 \end{array} \right) \\
ff \in \mathcal{C}[\![e_1 \ relop \ e_2]\!](m) & \stackrel{def}{=} & \exists v_1, v2. \left(\begin{array}{l} v_1 \in \mathcal{E}[\![e_1]\!](m) \\ v_2 \in \mathcal{E}[\![\ e_2]\!](m) \\ \neg(v_1 \ relop \ v2) \end{array} \right) \\
be_1 \ \wedge \ be_2 \in \mathcal{C}[\![be_1 \ \mathbf{and} \ be_2]\!](m) & \stackrel{def}{=} & and \left(\begin{array}{l} be_1 \in \mathcal{C}[\![be_1]\!](m) \\ be_2 \in \mathcal{C}[\![be_2]\!](m) \end{array} \right)
\end{array}
$$

- $(x := e, m) \longrightarrow m[x \mapsto v]$, **where** $v \in \mathcal{E}[\![e]\!](m)$
- $(skip, m) \longrightarrow m$
- **If** $(S_1, m) \longrightarrow m'$, **then** $(S_1; S_2, m) \longrightarrow (S_2, m')$.
- **If** $tt \in \mathcal{C}[\![be]\!]$, **then** $(\mathbf{if}\ be\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end\ if}, m) \longrightarrow (S_1, m)$.
- **If** $ff \in \mathcal{C}[\![be]\!]$, **then**
  $(\mathbf{if}\ be\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end\ if}, m) \longrightarrow (S_2, m)$.
- **If** $tt \in \mathcal{C}[\![be]\!]$, **then**
  $(\mathbf{while}\ be\ \mathbf{do}\ S\ \mathbf{end\ do}, m) \longrightarrow (S; \mathbf{while}\ be\ \mathbf{do}\ S\ \mathbf{end\ do}, m)$.
- **If** $ff \in \mathcal{C}[\![be]\!]$, **then** $(\mathbf{while}\ be\ \mathbf{do}\ S\ \mathbf{end\ do}, m) \longrightarrow m$.

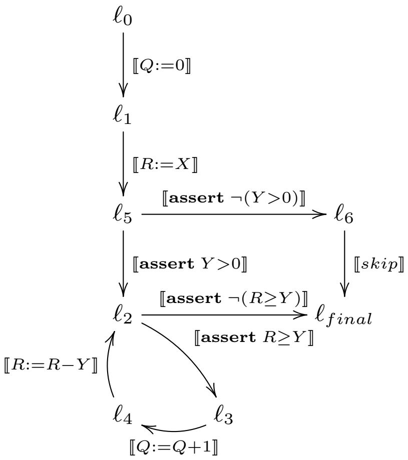
Generating Control Flowchart Graph from Program

- A *control flow graph* is generated from the program under consideration namely P.
- A control flow graph $\mathcal{CFG}[\![P]\!]$ is defined by nodes $(l \in \mathcal{C})$ which are program control points of P, $\mathcal{C}ontrol[\![P]\!]$ and by labelled edges with actions ($\mathcal{A}ctions[\![P]\!]$) defined by the following rules :

$$\begin{array}{lll} actions & ::= & v := exp \\ & | & skip \\ & | & \textbf{assert}\ be \end{array}$$

- A *control flow graph* is effectively defined by :
  - $\ell_{init} \in \mathcal{C}ontrol[\![P]\!]$: the entry point
  - $\ell_{end} \in \mathcal{C}ontrol[\![P]\!]$: the exit point
  - $\mathcal{E}dges[\![P]\!] \subseteq \mathcal{C}ontrol[\![P]\!] \times \mathcal{A}ctions[\![P]\!] \times \mathcal{C}ontrol[\![P]\!]$
- $\mathcal{CFG}[\![P]\!] = (\ell_{init}, \mathcal{E}dges[\![P]\!], \ell_{end})$

$\ell_0[Q := 0];$
$\ell_1[R := X];$
IF $\ell_5[Y > 0]$
  WHILE $\ell_2[R \geq Y]$
    $\ell_3[Q := Q{+}1];$
    $\ell_4[R := R{-}Y]$
  ENDWHILE
ELSE
  $\ell_6[skip]$
ENDIF



- $Mem \stackrel{def}{=} \mathbb{V} \longrightarrow \mathbb{Z}$
- Semantics of actions : $\stackrel{a}{\longrightarrow} \subseteq Mem \times Mem$
  $m \stackrel{x:=e}{\longrightarrow} m[x \mapsto v]$ **if** *there is a value* $v \in \mathcal{E}[\![e]\!](m)$
  $m \stackrel{skip}{\longrightarrow} m$
  $m \stackrel{\textbf{assert}\ be}{\longrightarrow} m]$ $if\ tt \in \mathcal{C}[\![be]\!](m)$
- Semantics for $\mathcal{CFG}[\![P]\!]$: $\stackrel{P}{\longrightarrow} \subseteq States \times States$
  - **If** $m \stackrel{a}{\longrightarrow} m'$ and $(\ell_1, a, \ell_2) \in \mathcal{E}dges[\![P]\!]$, then $(\ell_1, m) \stackrel{P}{\longrightarrow} (\ell_2, m')$
  - The set of initial states is $\{\ell_{init}\} \times Mem$
  - The set of reachable states for P is denoted $\text{REACHABLE}(P)$ and defined by $[\![P]\!] = \{s | \exists s_0 \in \{\ell_{init} \times Mem : s_0 \xrightarrow[\star]{P} s\}$.

- Defining for each control point ℓ of P the set of reachables values :
$$[\![P]\!]_{\ell}^{coll} = \{s | s \in States \wedge s \in [\![P]\!] \wedge \exists m \in Mem : s = (\ell, m)\}$$
- Characterizing $[\![P]\!]_{\ell}^{coll}$: it satisfies the system of equations
$$\forall \ell \in \mathcal{C}(P). X_{\ell} = X_{\ell}^{init} \cup \bigcup_{(\ell_1, a, \ell) \in \mathcal{E}dges[\![P]\!]} [\![a]\!](X_{\ell_1}) \tag{1}$$
- Let $a \in \mathcal{A}ctions[\![P]\!]$ and $x \subseteq Mem$.
$$[\![a]\!](x) = \{e | e \in States \wedge \exists f. f \in x \wedge f \xrightarrow{a} e\}$$
- $\forall \ell \in \mathcal{C}(P). \left(\begin{array}{l} \ell = \ell_{init} \Rightarrow X_{\ell}^{init} = Mem \\ \ell \neq \ell_{init} \Rightarrow X_{\ell}^{init} = \varnothing \end{array} \right)$

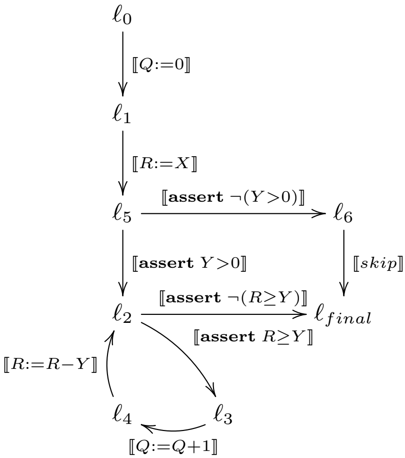
. . . .

☺ Théorème Let F the function defined as follows :

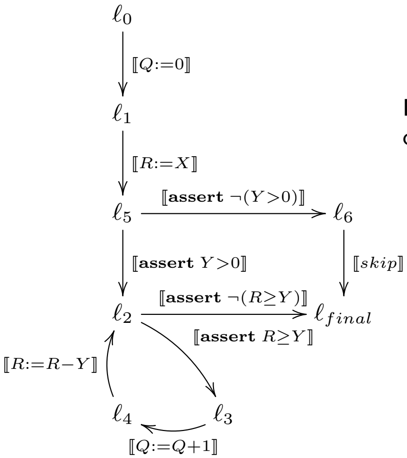
- n is the cardinality of $\mathcal{C}(P)$.
- $F \in \mathcal{P}(States)^n \longrightarrow \mathcal{P}(States)^n$
- If $X \in \mathcal{P}(States)^n$, then $F(X) = (\ldots, F_\ell(X), \ldots)$
- $\forall \ell \in \mathcal{C}(P).F_\ell(X) = X_\ell^{init} \cup \bigcup_{(\ell_1,a,\ell) \in \mathcal{E}dges[\![P]\!]} [\![a]\!](X_{\ell_1})$

The function F is monotonic over the complete lattice $(\mathcal{P}(States)^n, \subseteq)$ and has a least fixed-point μF defining the collecting semantics.

From flowchart to equational system

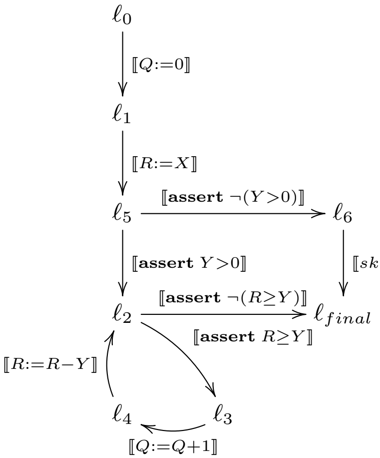


From flowchart to equational system



Defining equational systems for the collecting semantics :

From flowchart to equational system



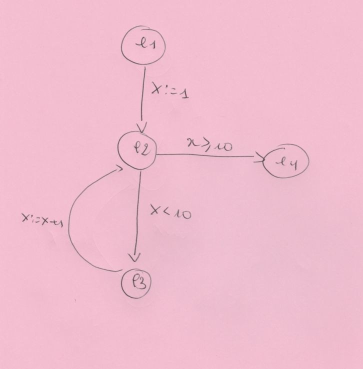
Defining equational systems for the collecting semantics :

$$
\begin{cases}
X_0 = Mem \\
X_1 = [\![Q := 0]\!](X_0) \\
X_5 = [\![R := X]\!](X_1) \\
X_2 = [\![assert\ (Y > 0)]\!](X_5) \cup [\![R := R{-}Y]\!](X_4) \\
X_3 = [\![assert\ R \geq Y]\!](X_2) \\
X_4 = [\![Q := Q{+}1]\!](X_3) \\
X_6 = [\![assert\ \neg(Y > 0)]\!](X_5) \\
X_7 = X_6 \cup [\![assert\ \neg(R \geq Y)]\!](X_2)
\end{cases}
$$

- The collecting semantics is the least fixed-point of the system of equations, which exists by fixed-point theorems.
- Questions :
  - How to compute the solution ?
  - Computing over finite structures, when it is possible....
  - Using an approximation of fixed-points ?
  - What is an approximation ?
  - What is an abstraction ?
  - What is the best abstraction ?

Next step

Defining a framework for computing lfp solution of these equational systems in any case.



- System of equations over $(\mathcal{P}(\mathbb{Z}, \subseteq)$
  - $X_1 = \mathbb{Z}$
  - $X_2 = \{1\} \cup \{v | v \in \mathbb{Z} \wedge v-1 \in X_3\}$
  - $X_3 = \{v | v \in X_2 \wedge v < 10\ \}$
  - $X_4 = \{v | v \in X_2 \wedge v \geq 10\ \}$
- Reachability
  - $X_1 = \mathbb{Z}$
  - $X_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
  - $X_3 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - $X_4 = \{10\}$