

MALG

Analyse des programmes

Année universitaire 2024-2025

- 1 Overview of the methodology
- 2 Standard, Collecting and Abstract Semantics

- ▶ **Syntax of programs** ($P \in PL$) defines the class of programs for applying the analysis@
- ▶ **Semantics** ($\llbracket P \rrbracket$) for each program ($P \in PL$)
- ▶

- 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.

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.

► *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 these properties

► Approximation

- Spaces of values should be restricted to computable entities
- Over-approximation of concrete properties

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

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
```


► Semantic Domains

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

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

► Semantics for Expressions

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

$$\mathcal{E}[[v]](m) \stackrel{def}{=} \{v\}$$

$$\mathcal{E}[[?]](m) \stackrel{def}{=} \mathbb{Z}$$

$$\mathcal{E}[[x]](m) \stackrel{def}{=} \{m(x)\}$$

$$\mathcal{E}[[e_1 \text{ op } e_2]](m) \stackrel{def}{=} \{v \mid \exists ve_1, ve_2. \left(\begin{array}{l} ve_1 \in \mathcal{E}[[e_1]](m) \\ ve_2 \in \mathcal{E}[[e_2]](m) \\ v = ve_1 \text{ o } ve_2 \end{array} \right) \}$$

► Semantics for conditions

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

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

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

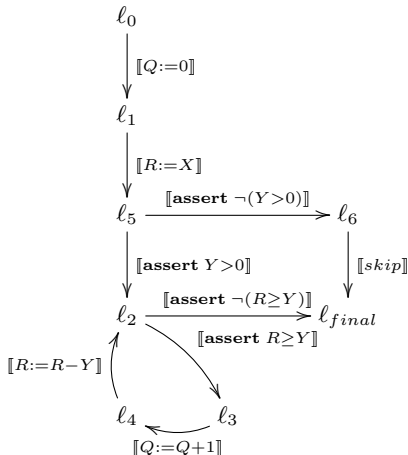
- ▶ A *control flow graph* is generated from the program under consideration namely P .
- ▶ A control flow graph $\mathcal{CFG}\llbracket P \rrbracket$ is defined by nodes ($l \in \mathcal{C}$) which are program control points of P , $\mathit{Control}\llbracket P \rrbracket$ and by labelled edges with actions ($\mathit{Actions}\llbracket P \rrbracket$) defined by the following rules :

$$\begin{array}{lcl} \mathit{actions} & ::= & v := \mathit{exp} \\ & | & \mathit{skip} \\ & | & \mathbf{assert} \ \mathit{be} \end{array}$$

- ▶ A *control flow graph* is effectively defined by :
 - $\ell_{init} \in \mathit{Control}\llbracket P \rrbracket$: the entry point
 - $\ell_{end} \in \mathit{Control}\llbracket P \rrbracket$: the exit point
 - $\mathcal{Edges}\llbracket P \rrbracket \subseteq \mathit{Control}\llbracket P \rrbracket \times \mathit{Actions}\llbracket P \rrbracket \times \mathit{Control}\llbracket P \rrbracket$
- ▶ $\mathcal{CFG}\llbracket P \rrbracket = (\ell_{init}, \mathcal{Edges}\llbracket P \rrbracket, \ell_{end})$

From program to flowchart

```
l0[Q := 0];  
l1[R := X];  
IF l5[Y > 0]  
    WHILE l2[R ≥ Y]  
        l3[Q := Q + 1];  
        l4[R := R - Y]  
    ENDWHILE  
ELSE  
    l6[skip]  
ENDIF
```



- ▶ $Mem \stackrel{def}{=} \mathbb{V} \longrightarrow \mathbb{Z}$
- ▶ Semantics of actions : $\xrightarrow{a} \subseteq Mem \times Mem$
 - $m \xrightarrow{x:=e} m[x \mapsto v]$ if there is a value $v \in \mathcal{E}[e](m)$
 - $m \xrightarrow{skip} m$
 - $m \xrightarrow{\text{assert } be} m$ if $tt \in \mathcal{C}[be](m)$
- ▶ Semantics for $\mathcal{CFG}[P]$: $\xrightarrow{P} \subseteq States \times States$
 - If $m \xrightarrow{a} m'$ and $(\ell_1, a, \ell_2) \in \mathcal{Edges}[P]$, then $(\ell_1, m) \xrightarrow{P} (\ell_2, m')$
 - 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 $\llbracket P \rrbracket = \{s \mid \exists s_0 \in \{\ell_{init}\} \times Mem : s_0 \xrightarrow[\star]{P} s\}$.

- ▶ Defining for each control point ℓ of P the set of reachable values :

$$\llbracket P \rrbracket_{\ell}^{coll} = \{s \mid s \in States \wedge s \in \llbracket P \rrbracket \wedge \exists m \in Mem : s = (\ell, m)\}$$

- ▶ Characterizing $\llbracket P \rrbracket_{\ell}^{coll}$: it satisfies the system of equations

$$\forall \ell \in \mathcal{C}(P). X_{\ell} = X_{\ell}^{init} \cup \bigcup_{(\ell_1, a, \ell) \in \mathcal{E}dges[P]} \llbracket a \rrbracket(X_{\ell_1}) \quad (1)$$

- ▶ Let $a \in \mathcal{A}ctions[\llbracket P \rrbracket]$ and $x \subseteq Mem$.

$$\llbracket a \rrbracket(x) = \{e \mid e \in States \wedge \exists f. f \in x \wedge f \xrightarrow{a} e\}$$

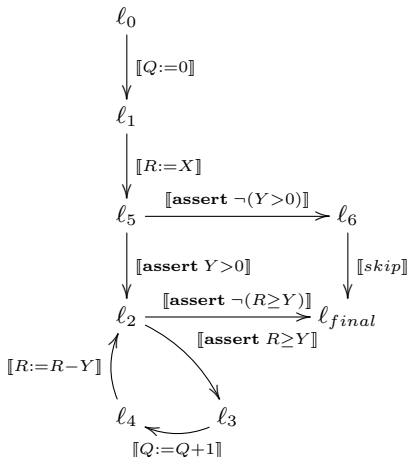
- ▶ $\forall \ell \in \mathcal{C}(P). \left(\begin{array}{l} \ell = \ell_{init} \Rightarrow X_{\ell}^{init} = Mem \\ \ell \neq \ell_{init} \Rightarrow X_{\ell}^{init} = \emptyset \end{array} \right)$

.....
☺ Théorème Let F the function defined as follows :

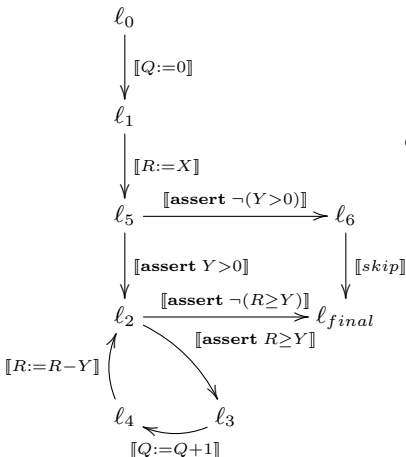
- ▶ n is the cardinality of $\mathcal{C}(P)$.
- ▶ $F \in \mathcal{P}(\text{States})^n \longrightarrow \mathcal{P}(\text{States})^n$
- ▶ If $X \in \mathcal{P}(\text{States})^n$, then $F(X) = (\dots, F_\ell(X), \dots)$
- ▶ $\forall \ell \in \mathcal{C}(P). F_\ell(X) = X_\ell^{init} \cup \bigcup_{(\ell_1, a, \ell) \in \text{Edges}[[P]]} \llbracket a \rrbracket(X_{\ell_1})$

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

From flowchart to equational system

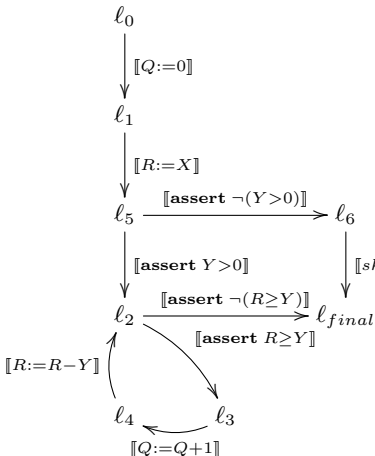


From flowchart to equational system



Defining equational systems for the collecting semantics :

From flowchart to equational system



Defining equational systems for the collecting semantics :

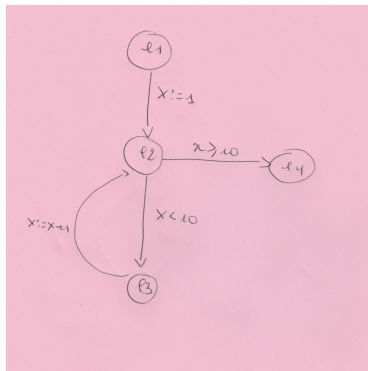
$$\left\{ \begin{array}{l} X_0 = Mem \\ X_1 = \llbracket Q := 0 \rrbracket(X_0) \\ X_5 = \llbracket R := X \rrbracket(X_1) \\ X_2 = \llbracket \text{assert } (Y > 0) \rrbracket(X_5) \cup \llbracket R := R - Y \rrbracket(X_4) \\ X_3 = \llbracket \text{assert } R \geq Y \rrbracket(X_2) \\ X_4 = \llbracket Q := Q + 1 \rrbracket(X_3) \\ X_6 = \llbracket \text{assert } \neg(Y > 0) \rrbracket(X_5) \\ X_7 = X_6 \cup \llbracket \text{assert } \neg(R \geq Y) \rrbracket(X_2) \end{array} \right.$$

- ▶ 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.

Example for computing reachable states



► System of equations over $(\mathcal{P}(\mathbb{Z}, \subseteq)$

- $X_1 = \mathbb{Z}$
- $X_2 = \{1\} \cup \{v \mid v \in \mathbb{Z} \wedge v-1 \in X_3\}$
- $X_3 = \{v \mid v \in X_2 \wedge v < 10\}$
- $X_4 = \{v \mid v \in X_2 \wedge v \geq 10\}$

► Reachability

- $X_1 = \mathbb{Z}$
- $X_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
- $X_3 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $X_4 = \{10\}$