Analyse statique de programmes

CentraleSupélec – année 2024/2025

Virgile Prevosto
virgile.prevosto@cea.fr

cours 2 – 21 mars 2025

CEA List
Laboratoire de Sûreté et Sécurité des Logiciels

Langage While

Syntaxe du langage While

Expressions *Expr*

e	::=	x	variable, $x \in \mathit{Var}$
	|	z	constante entière, $z \in \mathbb{Z}$
	|	$e + e \mid - e \mid e * e \mid e \,/\, e$	expressions arithmétiques
	|	$e = e \mid e <= e \mid \,!\, e$	expressions booléennes
	|	(e)	

Instructions *Instr*

i	::=	x := e	affectation
	|	if e then i else i fi	conditionnelle
	|	while e do i done	boucle
	|	$i ; i$	séquence
	|	skip	sans effet

Sémantique opérationnelle

Une sémantique opérationnelle est définie par un système de règles indiquant comment calculer les expressions et les instructions du langage.

Sémantique à grand pas

Une sémantique opérationnelle est dite à grand pas : elle décrit comment calculer le résultat final.

Sémantique à petit pas

Une sémantique opérationnelle à petit pas décrit comment effectuer chaque étape de calcul.

Valeurs

L'ensemble *Val* des valeurs est l'ensemble des constantes entières.

Environnement d'évaluation

Un environnement (d'évaluation) ρ est une fonction partielle de *Var* dans *Val*. On note $\rho[x \mapsto z]$ l'environnement ρ' tel que, pour toute variable y :

$$\rho'(y) \triangleq \begin{cases} z & \text{si } x = y \\ \rho(y) & \text{sinon et si } y \in \mathrm{dom}(\rho). \end{cases}$$

On note Env_ϵ l'ensemble des environnements : $Env_\epsilon \triangleq Var \rightharpoonup Val$.

Jugement d'évaluation

On note $\rho \vdash e \Downarrow_\epsilon z$ (resp. $\rho \vdash i \Downarrow \rho'$) l'évaluation de l'expression e (resp. l'instruction i) dans l'environnement ρ conduisant à la valeur $z \in Val$ (resp. l'environnement ρ').

Évaluation des expressions entières

$$\frac{x \in \mathrm{dom}(\rho) \qquad \rho(x) = z}{\rho \vdash x \Downarrow_\epsilon z} \qquad \frac{}{\rho \vdash z \Downarrow_\epsilon z} \qquad \frac{\rho \vdash e \Downarrow_\epsilon z \qquad z' = -z}{\rho \vdash -e \Downarrow_\epsilon z'}$$

$$\frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z' = z_1 + z_2}{\rho \vdash e_1 + e_2 \Downarrow_\epsilon z'}$$

$$\frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z' = z_1 / z_2 \qquad z_2 \neq 0}{\rho \vdash e_1 \,/\, e_2 \Downarrow_\epsilon z'}$$

La règle pour $*$ (avec $\times$) est similaire à celle de $+$ (avec $+$) et omise.

Évaluation des expressions booléennes

$$\frac{\rho \vdash e \Downarrow_\epsilon 0}{\rho \vdash \texttt{!}\, e \Downarrow_\epsilon 1} \qquad \frac{\rho \vdash e \Downarrow_\epsilon z \qquad z \neq 0}{\rho \vdash \texttt{!}\, e \Downarrow_\epsilon 0}$$

$$\frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z_1 = z_2}{\rho \vdash e_1 \texttt{=} e_2 \Downarrow_\epsilon 1}$$

$$\frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z_1 \neq z_2}{\rho \vdash e_1 \texttt{=} e_2 \Downarrow_\epsilon 0} \qquad \frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z_1 \leq z_2}{\rho \vdash e_1 \texttt{<=} e_2 \Downarrow_\epsilon 1}$$

$$\frac{\rho \vdash e_1 \Downarrow_\epsilon z_1 \qquad \rho \vdash e_2 \Downarrow_\epsilon z_2 \qquad z_1 > z_2}{\rho \vdash e_1 \texttt{<=} e_2 \Downarrow_\epsilon 0}$$

Évaluation des instructions

$$\frac{}{\rho \vdash \text{skip} \Downarrow \rho} \qquad \frac{\rho \vdash e \Downarrow_\epsilon z}{\rho \vdash x := e \Downarrow \rho[x \mapsto z]} \qquad \frac{\rho \vdash e \Downarrow_\epsilon 0 \qquad \rho \vdash i_2 \Downarrow \rho_2}{\rho \vdash \text{if } e \text{ then } i_1 \text{ else } i_2 \text{ fi} \Downarrow \rho_2}$$

$$\frac{\rho \vdash e \Downarrow_\epsilon z \qquad z \neq 0 \qquad \rho \vdash i_1 \Downarrow \rho_1}{\rho \vdash \text{if } e \text{ then } i_1 \text{ else } i_2 \text{ fi} \Downarrow \rho_1} \qquad \frac{\rho \vdash i_1 \Downarrow \rho_1 \qquad \rho_1 \vdash i_2 \Downarrow \rho_2}{\rho \vdash i_1 ; i_2 \Downarrow \rho_2}$$

$$\frac{\rho \vdash \text{if } e \text{ then } i ; \text{while } e \text{ do } i \text{ done else skip} \Downarrow \rho'}{\rho \vdash \text{while } e \text{ do } i \text{ done} \Downarrow \rho'}$$

Exemple

$$\dfrac{\dfrac{}{\emptyset \vdash 2 \Downarrow_\epsilon 2}}{\emptyset \vdash x := 2 \Downarrow \rho_1} \qquad \dfrac{\dfrac{\dfrac{x \in \mathrm{dom}(\rho_1)}{\rho_1(x) = 2}}{\rho_1 \vdash x \Downarrow_\epsilon 2} \qquad \dfrac{}{\rho_1 \vdash 3 \Downarrow_\epsilon 3} \qquad 6 = 2 \times 3}{\rho_1 \vdash x * 3 \Downarrow_\epsilon 6}}{\rho_1 \vdash x := x * 3 \Downarrow \rho_2}$$

$$\emptyset \vdash x := 2; x := x * 3 \Downarrow \rho_2$$

Notations :

- $\rho_1 = \{x \mapsto 2\}$
- $\rho_2 = \{x \mapsto 6\}$

Remarques

Déterminisme

Cette sémantique est déterministe : au plus une règle est applicable à chaque fois.

Programmes erronés et ne terminant pas

- un programme contenant une erreur (e.g. division par zéro) n'a pas d'état résultant (on pourrait les écrire, mais c'est très lourd)
- un programme ne terminant pas non plus
- ils sont indistingables

Pro and cons

- proche d'une implémentation déclarative (OCaml, Prolog)
- erreurs et non terminaison ne sont pas distingables

Traces d'exécution

Définition

À un programme P, on associe un Graphe de flot de contrôle étiqueté $(\mathcal{S}, \mathcal{T}, \mathcal{E}, \epsilon, \mathcal{I}, \mathcal{F})$*, où :*

- *À chaque instruction de P on associe deux nœuds de* $\mathcal{S}$*, représentant respectivement l'état avant et après l'instruction.*
- $\mathcal{T} \subset (\mathcal{S} \times \mathcal{S})$ *est l'ensemble des arcs*
- $\mathcal{E} : \mathcal{T} \mapsto \{x := e \mid ?e \mid \text{skip} \mid \mathrm{Err}(e)\}$ *associe une étiquette à chaque arc.*
- $\mathcal{I} \in \mathcal{S}$ *représente l'état initial*
- $\mathcal{F} \in \mathcal{S}$ *représente l'état final*
- $\epsilon \in \mathcal{S}$ *représente l'état d'erreur*

Exemple

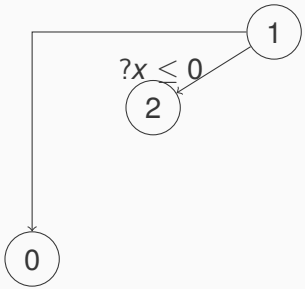
1. if $x \leq 0$ then
2. $x := 1$
3. else
4. skip
5. fi;
6. $y := 1$;
7. while ! $(x <= 0)$ do
8. $y := y \times x$;
9. $x := x - 1$;
10. done
11.

Exemple

1. if $x \leq 0$ then
2. $x := 1$
3. else
4. skip
5. fi;
6. $y := 1$;
7. while ! $(x <= 0)$ do
8. $y := y \times x$;
9. $x := x - 1$;
10. done
11.

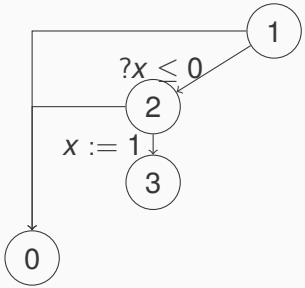
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



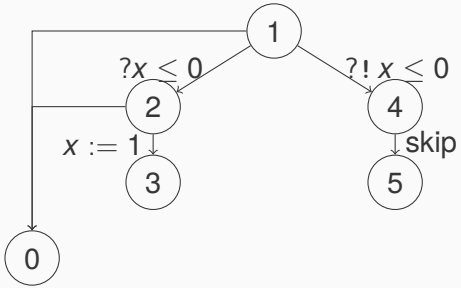
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



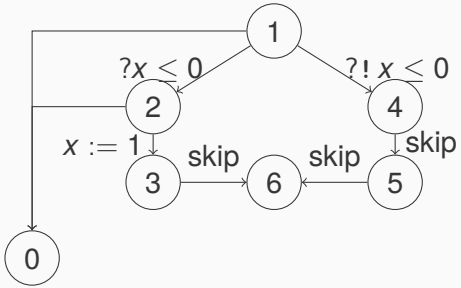
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



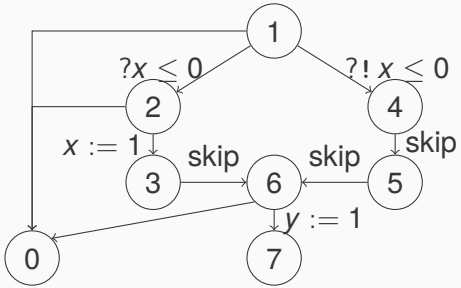
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



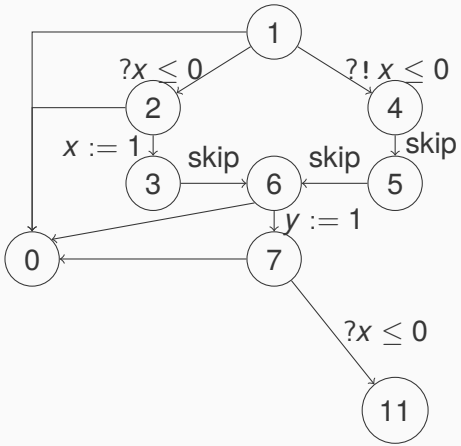
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



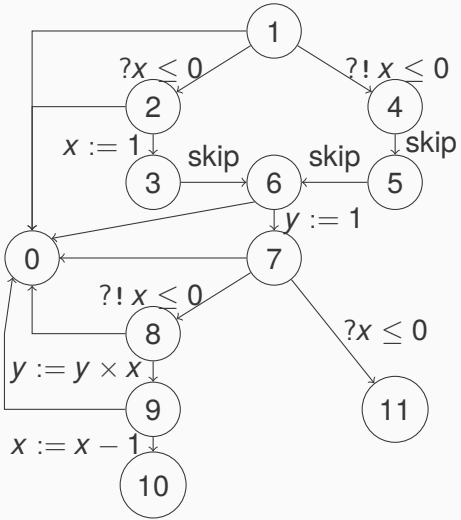
Exemple

1. if $x \leq 0$ then
2. $x := 1$
3. else
4. skip
5. fi;
6. $y := 1$;
7. while ! $(x <= 0)$ do
8. $y := y \times x$;
9. $x := x - 1$;
10. done
11.



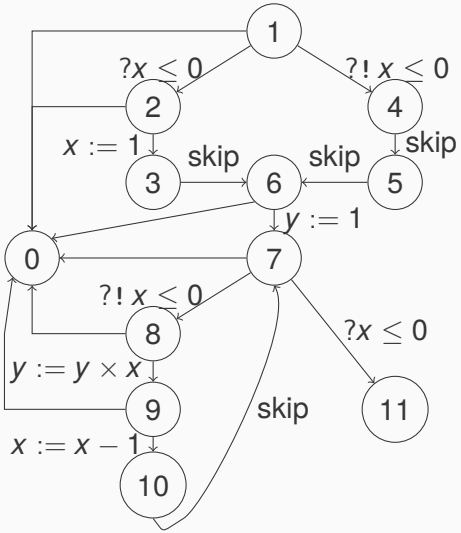
Exemple

1 if $x \leq 0$ then
2 $x := 1$
3 else
4 skip
5 fi;
6 $y := 1$;
7 while ! $(x <= 0)$ do
8 $y := y \times x$;
9 $x := x - 1$;
10 done
11



Exemple

1. if $x \leq 0$ then
2. $x := 1$
3. else
4. skip
5. fi;
6. $y := 1$;
7. while ! $(x <= 0)$ do
8. $y := y \times x$;
9. $x := x - 1$;
10. done
11.



Environnements et étiquettes

Skip

$$\frac{}{\rho^\flat \rightarrow^\flat \mathsf{skip} \rho^\flat}$$

Assignement

$$\frac{\rho^\flat \vdash e \Downarrow_\epsilon \mathrm{v}}{\rho^\flat \rightarrow^\flat x{:=}e \rho^\flat[x \mapsto v]}$$

Cond

$$\frac{\rho^\flat \vdash e \Downarrow_\epsilon \mathrm{v} \qquad \mathrm{v} \neq 0}{\rho^\flat \rightarrow^\flat ?e \rho^\flat}$$

Err

$$\frac{\forall \mathrm{v}, \neg \rho^\flat \vdash e \Downarrow_\epsilon \mathrm{v}}{\rho^\flat \rightarrow^\flat \mathit{Err}(e) \rho^\flat}$$

Définition

Un état concret du programme P est un couple $(s, \rho^\flat) \in \mathcal{S} \times Env_\epsilon$.
L'ensemble des transitions de P est une relation sur les états, $\Rightarrow^\flat_P$ *défini par*

$$(s_1, \rho_1) \Rightarrow^\flat_P (s_2, \rho_2) \Leftrightarrow (s_1, s_2) \in \mathcal{T} \wedge \rho_1 \rightarrow^\flat_{(s_1,s_2)} \rho_2$$

Trace d'exécution

Une trace d'exécution T d'un programme P est une suite (finie ou non) :

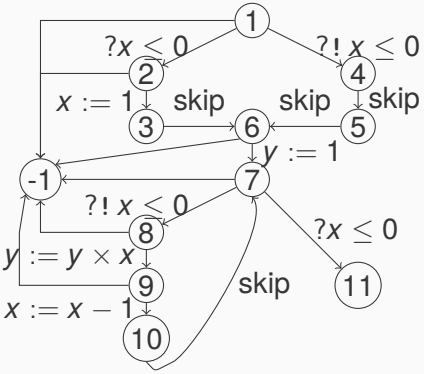
$$(s_0, \rho_0), (s_1, \rho_1), \cdots, (s_i, \rho_i), \cdots$$

telle que

- $s_0 = \mathcal{I}$ est le point d'entrée du programme
- $\forall i \geq 0 \quad (s_i, \rho_i) \Rightarrow^\flat_P (s_{i+1}, \rho_{i+1})$ est dérivable à partir des règles précédentes

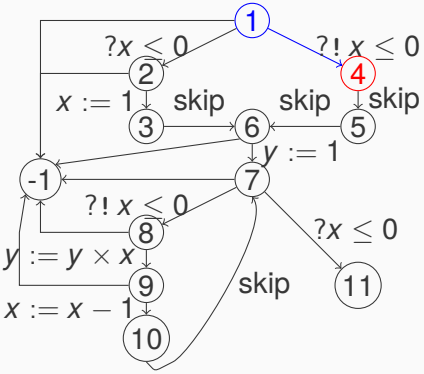
On note $\mathbb{T}(P)$ l'ensemble des traces d'exécution du programme P.

Exemple de trace



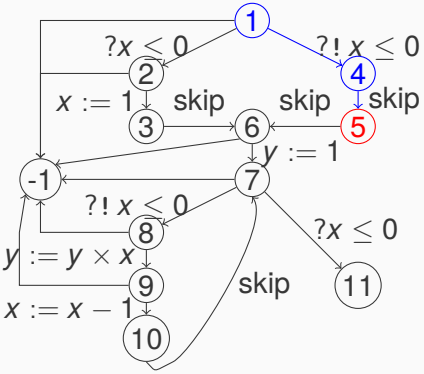
Exemple de trace

$(s_1, \{x \mapsto 1\})$

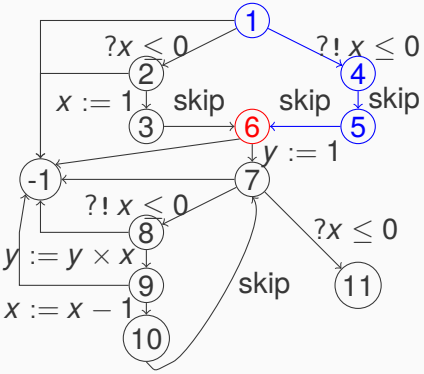


Exemple de trace

$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$



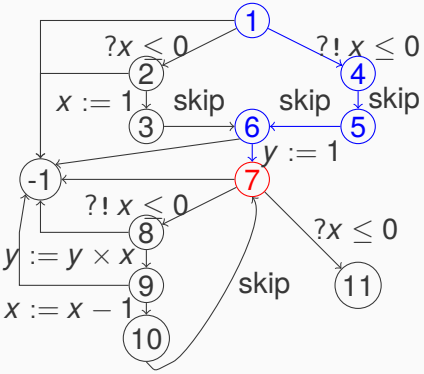
Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$

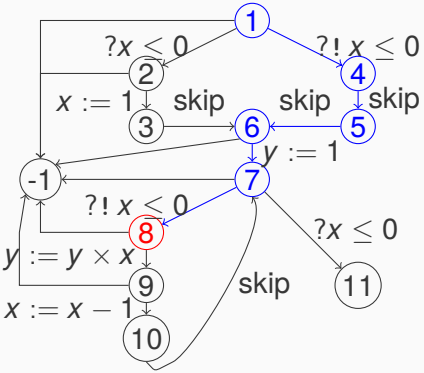
Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

Exemple de trace

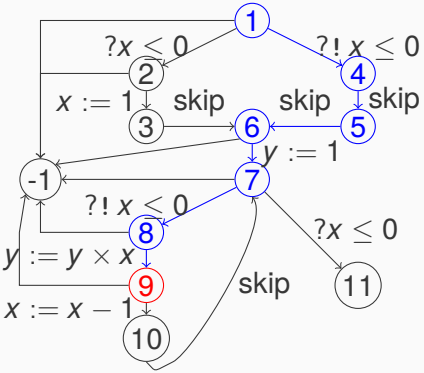


$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

$(s_7, \{x \mapsto 1; y \mapsto 1\})$

Exemple de trace



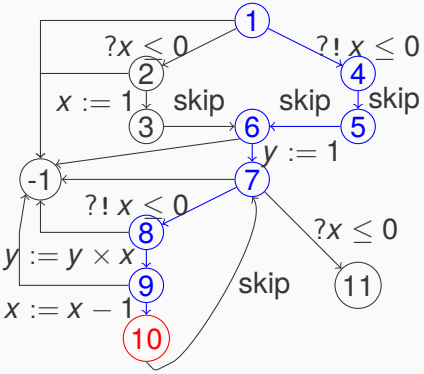
$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

$(s_7, \{x \mapsto 1; y \mapsto 1\})$

$(s_8, \{x \mapsto 1, y \mapsto 1\})$

Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

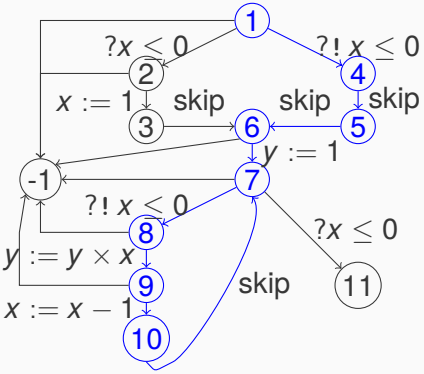
$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

$(s_7, \{x \mapsto 1; y \mapsto 1\})$

$(s_8, \{x \mapsto 1, y \mapsto 1\})$

$(s_9, \{x \mapsto 1, y \mapsto 1\})$

Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

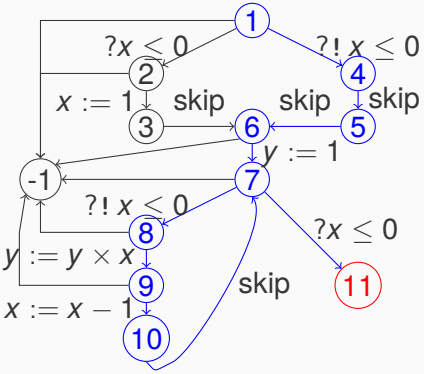
$(s_7, \{x \mapsto 1; y \mapsto 1\})$

$(s_8, \{x \mapsto 1, y \mapsto 1\})$

$(s_9, \{x \mapsto 1, y \mapsto 1\})$

$(s_{10}, \{x \mapsto 0, y \mapsto 1\})$

Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

$(s_7, \{x \mapsto 1; y \mapsto 1\})$

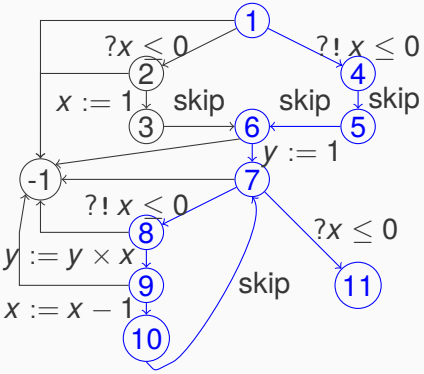
$(s_8, \{x \mapsto 1, y \mapsto 1\})$

$(s_9, \{x \mapsto 1, y \mapsto 1\})$

$(s_{10}, \{x \mapsto 0, y \mapsto 1\})$

$(s_7, \{x \mapsto 0, y \mapsto 1\})$

Exemple de trace



$(s_1, \{x \mapsto 1\})$ $(s_4, \{x \mapsto 1\})$

$(s_5, \{x \mapsto 1\})$ $(s_6, \{x \mapsto 1\})$

$(s_7, \{x \mapsto 1; y \mapsto 1\})$

$(s_8, \{x \mapsto 1, y \mapsto 1\})$

$(s_9, \{x \mapsto 1, y \mapsto 1\})$

$(s_{10}, \{x \mapsto 0, y \mapsto 1\})$

$(s_7, \{x \mapsto 0, y \mapsto 1\})$

$(s_{11}, \{x \mapsto 0, y \mapsto 1\})$

Problèmes

- L'ensemble des traces est infini
- Existence de traces infinies
- Approximation de la sémantique de traces se concentrant sur des propriétés particulières
  - Analyse flot de données
  - Correction ?
  - Terminaison ?
- En particulier, rassembler toutes les traces possibles :
  - Merge Over all Paths,
  - Sémantique Collectrice

Analyse statique

Treillis complet

$(E, \sqsubseteq)$ est un treillis complet si et seulement si

- $\sqsubseteq$ est un ordre partiel
- $\forall F \subseteq E$, $\{x | \forall y \in F, x \sqsubseteq y\}$ possède un plus grand élément, $\sqcap F$
- $\forall F \subseteq E$, $\{x | \forall y \in F, y \sqsubseteq x\}$ possède un plus petit élément, $\sqcup F$
- $\sqcap E = \sqcup \emptyset$ est noté $\bot$
- $\sqcup E = \sqcap \emptyset$ est noté $\top$

Chaînes

- Une chaîne est une séquence (x_i) ordonnée : $x_1 \sqsubseteq x_2 \sqsubseteq ...x_n \sqsubseteq ...$ (chaîne ascendante) ou $x_1 \sqsupseteq x_2 \sqsupseteq ...x_n \sqsupseteq ...$ (chaîne descendante)
- Un treillis complet vérifie la condition de chaîne ascendante (descendante) si et seulement si toute chaîne ascendante (descendante) est stationnaire : $\exists N, \forall i, j \geq N, x_i = x_j$

Théorème du point fixe de Tarski (1953)

Toute fonction monotone f sur un treillis complet $(E, \sqsubseteq)$ admet un plus petit point fixe, $\mathrm{lfp}(f) = f(\mathrm{lfp}(f))$. De plus :

$$\mathrm{lfp}(f) = \sqcap\{x | f(x) \sqsubseteq x\}$$

Itération de Kleene

$$\begin{aligned} x_0 &= \bot \\ x_{n+1} &= f(x_n) \end{aligned}$$

- (x_i) est croissante
- Si elle converge, elle atteint $\mathrm{lfp}(f)$.
- C'est le cas si E vérifie la condition de chaîne ascendante.

Analyse flot de données (data flow)

Vue informelle

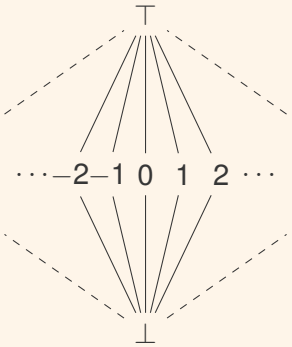
- on propage des informations le long du graphe de contrôle (fonction de transition)
- lorsqu'un sommet admet plusieurs prédécesseurs (sommet de jonction), on considère la réunion des informations propagés
- on termine lorsque la propagation des informations ne fait plus "augmenter" le résultat

Questions

- ce procédé termine-t-il toujours ?
- ce procédé est-il correct ?

Treillis de base

On part du treillis $\mathcal{C}$ suivant :



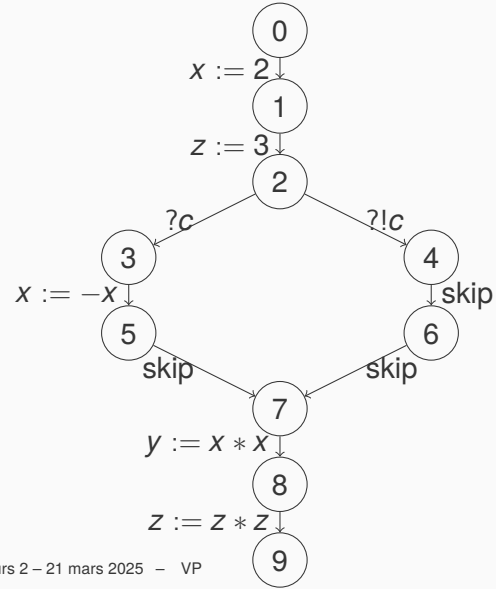
Cadre d'analyse

- $\Sigma = \mathit{Var} \rightharpoonup \mathcal{C}$
- Initialement : $\sigma_i = \lambda x.\top$
- $f_{x:=e}(\sigma^\sharp) = \sigma^\sharp[x \leftarrow [\![e]\!]_\sharp \sigma^\sharp]$
- $f_{\mathsf{skip}}(\sigma^\sharp) = \sigma^\sharp$
- $f_{?e}(\sigma^\sharp) = \begin{cases} \bot & \text{si } [\![e]\!]_\sharp \sigma^\sharp = 0 \\ \lfloor \sigma^\sharp \rfloor_e & \text{sinon} \end{cases}$

Propriétés

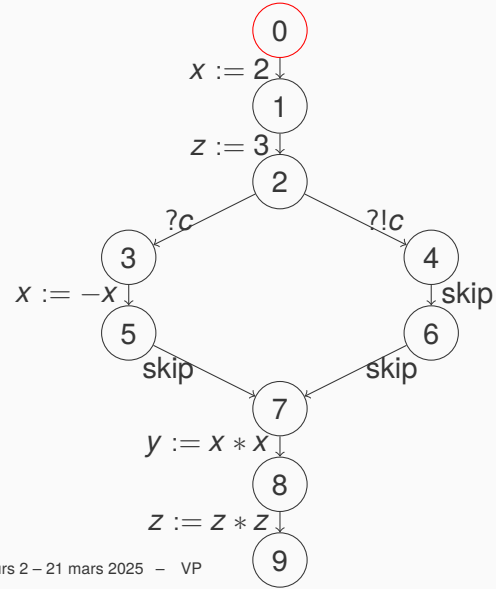
- Correction : Si à la fin de l'analyse, $\sigma^\sharp_i(x) = \mathrm{v}$, toute trace passant par i associe v à x.
- Terminaison : L'analyse termine toujours

Exemple de propagation



nœud	c	x	y	z
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				

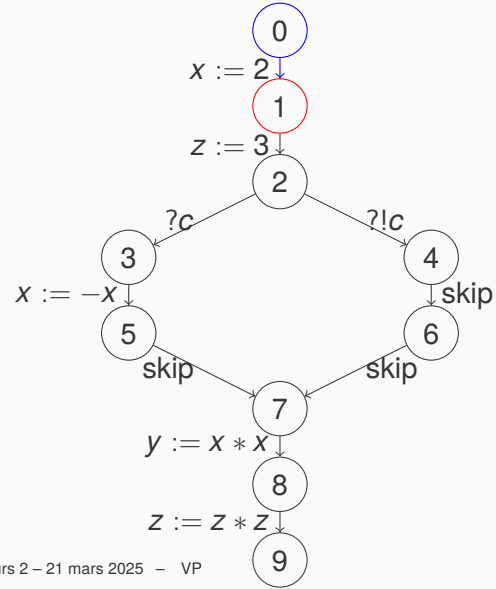
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1				
2				
3				
4				
5				
6				
7				
8				
9				

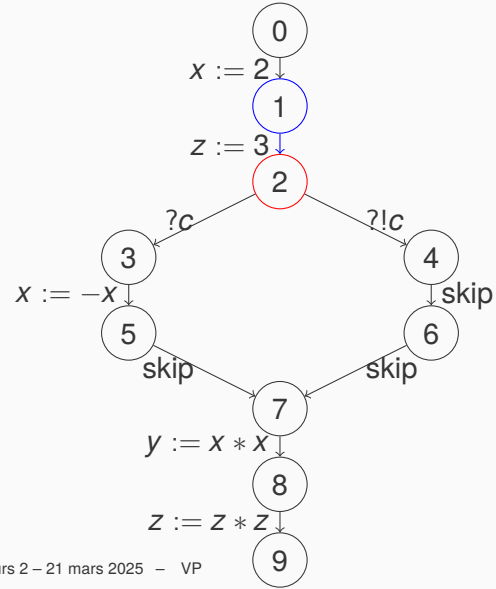
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2				
3				
4				
5				
6				
7				
8				
9				

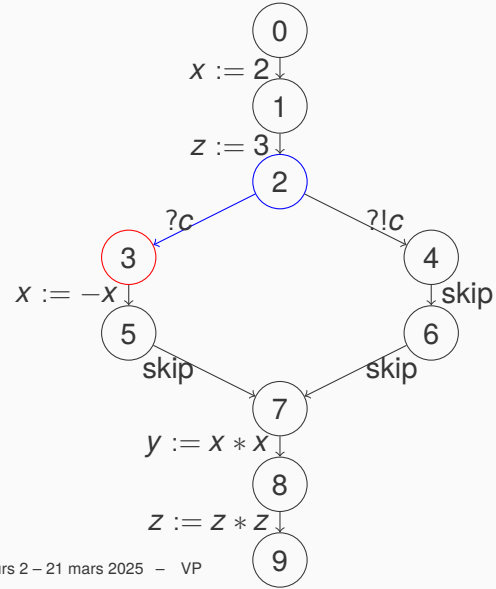
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3				
4				
5				
6				
7				
8				
9				

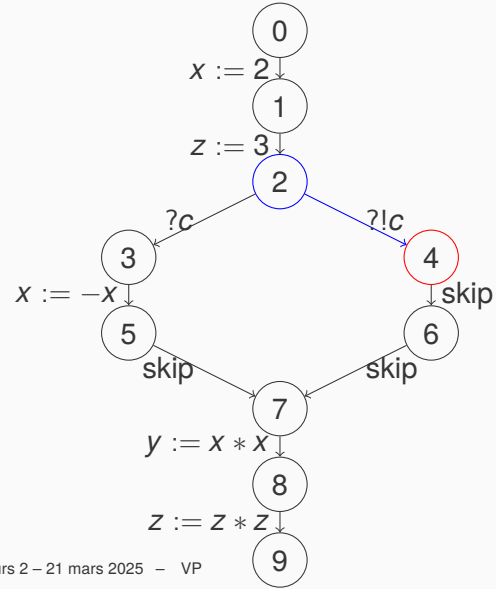
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4				
5				
6				
7				
8				
9				

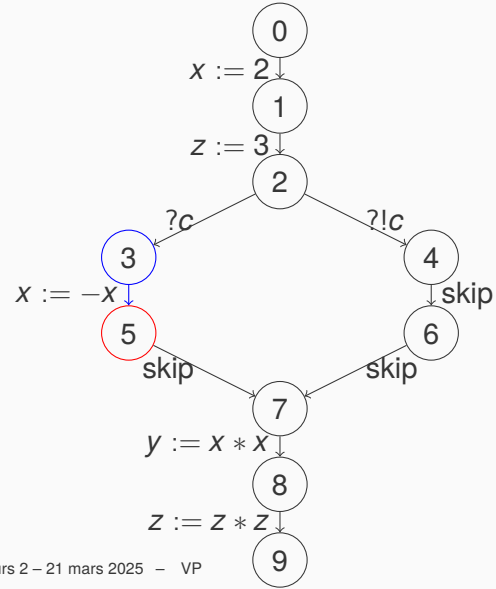
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5				
6				
7				
8				
9				

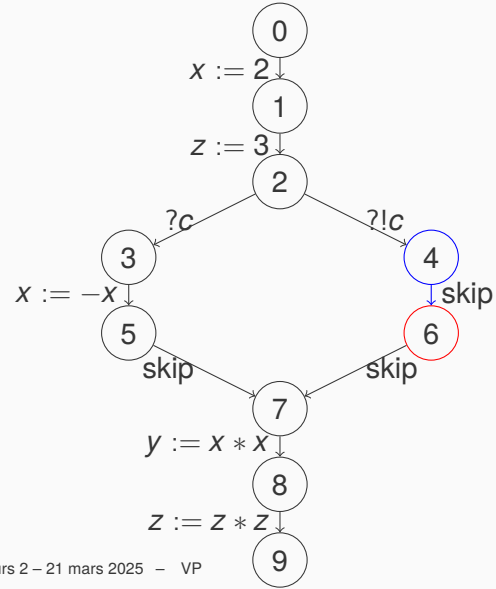
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6				
7				
8				
9				

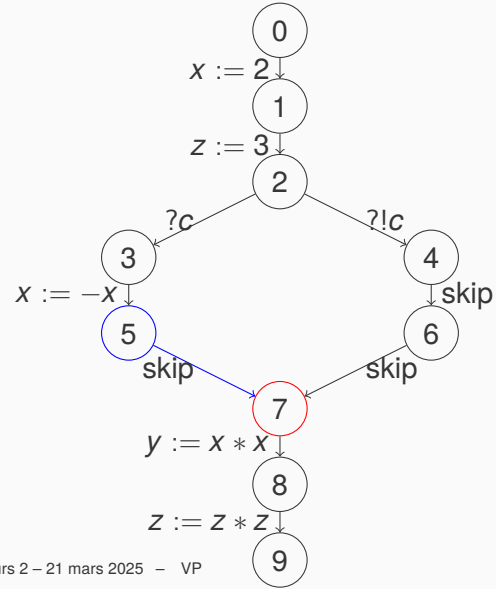
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6	0	2	⊤	3
7				
8				
9				

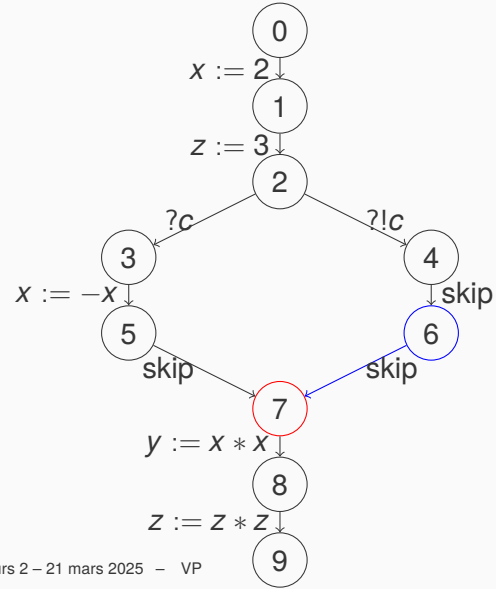
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6	0	2	⊤	3
7	⊤	−2	⊤	3
8				
9				

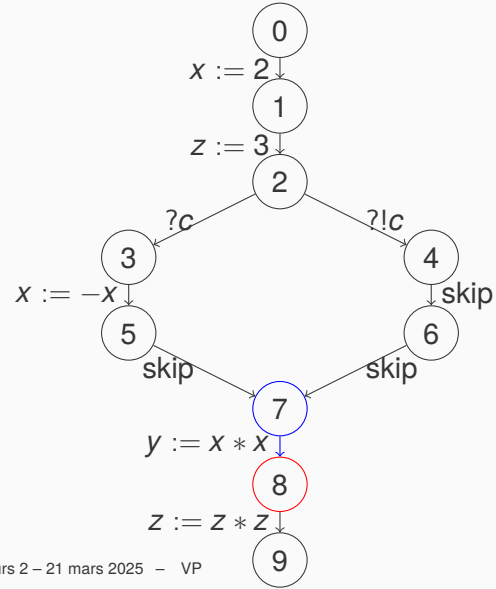
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6	0	2	⊤	3
7	⊤	⊤	⊤	3
8				
9				

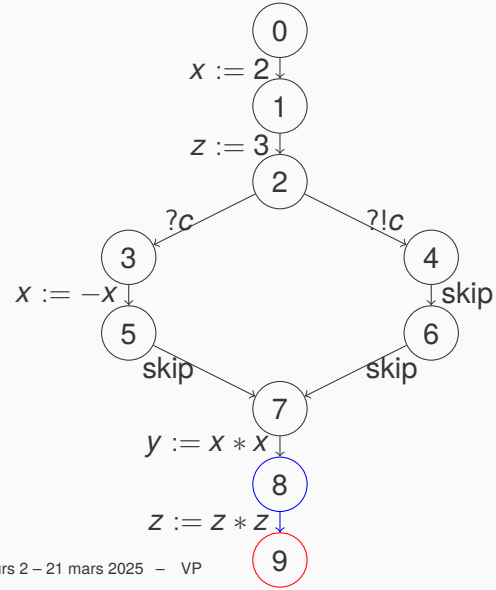
Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6	0	2	⊤	3
7	⊤	⊤	⊤	3
8	⊤	⊤	⊤	3
9				

Exemple de propagation



⊤ = "pas d'information"

nœud	c	x	y	z
0	⊤	⊤	⊤	⊤
1	⊤	2	⊤	⊤
2	⊤	2	⊤	3
3	⊤	2	⊤	3
4	0	2	⊤	3
5	⊤	−2	⊤	3
6	0	2	⊤	3
7	⊤	⊤	⊤	3
8	⊤	⊤	⊤	3
9	⊤	⊤	⊤	9

Cadre monotone

Un cadre monotone est un quadruplet $(\Sigma, \sqsubseteq, f, i)$ tel que :

- $(\Sigma, \sqsubseteq)$ un treillis complet d'états
- $f : \mathcal{E} \to \Sigma \to \Sigma$ un ensemble de fonctions de transitions
- $\forall e \in \mathcal{E}, f_e$ est monotone
- $i \in \Sigma$ l'état de départ de l'analyse

Remarque

Le cadre d'analyse défini pour la propagation de constantes est un cadre monotone.

Définition

Soit un cadre d'analyse $(\Sigma, \sqsubseteq, f, i)$, et un graphe $G = (\mathcal{S}, \mathcal{T}, \mathcal{E}, \epsilon, \mathcal{I}, \mathcal{F})$. L'instance du cadre sur G est une équation d'inconnue $\sigma : \mathcal{S} \to \Sigma$:

$$\sigma = \lambda s. \bigsqcup_{s' \in \mathrm{Pred}(s)} \{f_{\epsilon(s',s)}(\sigma(s'))\} \sqcup i_s$$

où

$$i_s = \begin{cases} i & \text{si } s = \mathcal{I} \\ \bot & \text{sinon} \end{cases}$$

Remarque

$\mathcal{S} \to \Sigma$ est un treillis complet ($f \sqsubseteq g \Leftrightarrow \forall s, f(s) \sqsubseteq g(s)$)

Notation

Soit F: $(\mathcal{S} \to \Sigma) \to \mathcal{S} \to \Sigma = \lambda\sigma.\lambda s.\bigsqcup_{s' \in \mathrm{Pred}(s)}\{f_{\epsilon(s',s)}(\sigma(s'))\} \sqcup i_s$.

Lemme

F est une fonction monotone sur $\mathcal{S} \to \Sigma$

Notation

Soit F: $(\mathcal{S} \to \Sigma) \to \mathcal{S} \to \Sigma = \lambda\sigma.\lambda s.\bigsqcup_{s' \in \mathrm{Pred}(s)}\{f_{\epsilon(s',s)}(\sigma(s'))\} \sqcup i_s$.

Lemme

F est une fonction monotone sur $\mathcal{S} \to \Sigma$

Lemme

Si Σ vérifie la condition de chaîne ascendante, et $\mathcal{S}$ est fini, alors $\mathcal{S} \to \Sigma$ aussi.

Notation

Soit F: $(\mathcal{S} \to \Sigma) \to \mathcal{S} \to \Sigma = \lambda\sigma.\lambda s. \bigsqcup_{s' \in \mathrm{Pred}(s)} \{f_{\epsilon(s',s)}(\sigma(s'))\} \sqcup i_s$.

Lemme

F est une fonction monotone sur $\mathcal{S} \to \Sigma$

Lemme

Si Σ vérifie la condition de chaîne ascendante, et $\mathcal{S}$ est fini, alors $\mathcal{S} \to \Sigma$ aussi.

Application

On peut donc utiliser l'itération de Kleene pour résoudre l'équation :

$$\sigma_0 = \bot$$

$$\sigma_{n+1} = \lambda s. \bigsqcup_{s' \in \mathrm{Pred}(s)} \{f_{\epsilon(s',s)}(\sigma_n(s'))\} \sqcup i_s$$

Complexité

La complexité de la méthode de Kleene est souvent assez mauvaise en pratique.

Itération chaotique

On peut exploiter le fait qu'on travaille sur un treillis particulier ($\mathcal{S} \to \Sigma$) pour proposer une autre méthode d'itération.

Soit s_j un sommet tel que $\sigma_n(s_j) \sqsubset F(\sigma_n)(s_j)$

$$\sigma_{n+1} = \lambda s. \begin{cases} \sigma_n(s) & \text{si } s \neq s_j \\ \bigsqcup_{s' \in \mathrm{Pred}(s_j)} \{f_{\epsilon(s',s_j)}(\sigma_n(s'))\} \sqcup i_{s_j} & \text{si } s = s_j \end{cases}$$

Intuition : on ne re-calcule que pour les successeurs de s_j (on propage les nouvelles informations)

Implémentation de l'analyse

1: **procedure** ChaoticIteration(f_e, i)
2: $\quad\sigma[\mathcal{I}] \leftarrow i$
3: $\quad$*WorkList* $\leftarrow \{(\mathcal{I}, s) | s \in \mathrm{Succ}(\mathcal{I})\}$
4: $\quad$**while** *WorkList* $\neq \emptyset$ **do**
5: $\quad\quad(s, s') \leftarrow$ *Choose*(*WorkList*)
6: $\quad\quad v \leftarrow f_{\epsilon(s,s')}(\sigma[s])\}$
7: $\quad\quad$**if** $v \sqsubseteq \sigma[s']$ **then**
8: $\quad\quad\quad$*WorkList* $\leftarrow$ *WorkList*$\backslash\{(s, s')\}$
9: $\quad\quad$**else**
10: $\quad\quad\quad\sigma[s'] \leftarrow \sigma[s'] \sqcup v$
11: $\quad\quad\quad$*WorkList* $\leftarrow$ *WorkList*$\backslash\{(s, s')\} \cup \{(s', s'') | s'' \in \mathrm{Succ}(s')\}$
12: $\quad\quad$**end if**
13: $\quad$**end while**
14: **end procedure**

Lemme : terminaison

Si Σ vérifie la condition de chaîne ascendante, l'itération chaotique termine

Propriétés de l'itération chaotique

Lemme : terminaison

Si Σ vérifie la condition de chaîne ascendante, l'itération chaotique termine

Lemme : correction

La fonction obtenue par itération chaotique est $\text{lfp}(F)$

Définition

A chaque point de programme s d'un programme P on associe l'ensemble :

$$\mathcal{C}_P(s) = \left\{ \begin{array}{l|ll} \rho^\flat \in Env^\flat & \exists (s_0,\rho_0),(s_1,\rho_1),\cdots,(s_i,\rho_i),\cdots \in \mathbb{T}(P) \\ & \exists k\ s = s_k \text{ et } \rho^\flat = \rho_k \end{array} \right\}$$

$$\mathcal{C}_P \in Env_P^\flat = \{f^\flat : \mathcal{S} \to \wp(Env^\flat)\}$$

Rappel

$Env_P^\flat$ peut être muni d'un ordre partiel :

$$\forall f_1^\flat, f_2^\flat \in Env_P^\flat, \quad f_1^\flat \subseteq f_2^\flat \qquad \text{ssi} \qquad \forall s \in \mathcal{S}, \quad f_1^\flat(s) \subseteq f_2^\flat(s).$$

On cherche à calculer la sémantique collectrice de P, $Coll_P$, comme une analyse flot de données.

- On se donne un ensemble d'états initiaux arbitraires $R_{\mathcal{I}}$
- Les fonctions de transition sont de la forme :

$$\mathcal{P}^\flat_e = \lambda R.\{\rho \mid \exists \rho' \in R, \rho' \rightarrow^\flat_e \rho\}$$

- On part de $\lambda s \in \mathcal{S}. \begin{cases} R_{\mathcal{I}} & \text{si } s = \mathcal{I} \\ \emptyset & \text{sinon} \end{cases}$ (qui appartient à $Env^\flat_P$)

On cherche à calculer la sémantique collectrice de P, $Coll_P$, comme une analyse flot de données.

- On se donne un ensemble d'états initiaux arbitraires $R_{\mathcal{I}}$
- Les fonctions de transition sont de la forme :

$$\mathcal{P}_e^\flat = \lambda R.\{\rho \mid \exists \rho' \in R, \rho' \rightarrow^\flat_e \rho\}$$

- On part de $\lambda s \in \mathcal{S}.\begin{cases} R_{\mathcal{I}} & \text{si } s = \mathcal{I} \\ \emptyset & \text{sinon} \end{cases}$ (qui appartient à $Env_P^\flat$)

Lemme

$\mathcal{P}_e^\flat$ est une fonction monotone

On cherche à calculer la sémantique collectrice de P, $Coll_P$, comme une analyse flot de données.

- On se donne un ensemble d'états initiaux arbitraires $R_{\mathcal{I}}$
- Les fonctions de transition sont de la forme :

$$\mathcal{P}_e^\flat = \lambda R.\{\rho \mid \exists \rho' \in R, \rho' \rightarrow^\flat_e \rho\}$$

- On part de $\lambda s \in \mathcal{S}. \begin{cases} R_{\mathcal{I}} & \text{si } s = \mathcal{I} \\ \emptyset & \text{sinon} \end{cases}$ (qui appartient à $Env_P^\flat$)

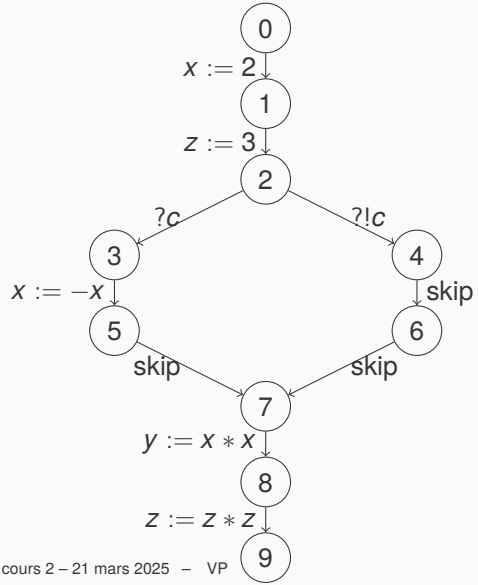
Lemme

$\mathcal{P}_e^\flat$ est une fonction monotone

Remarque

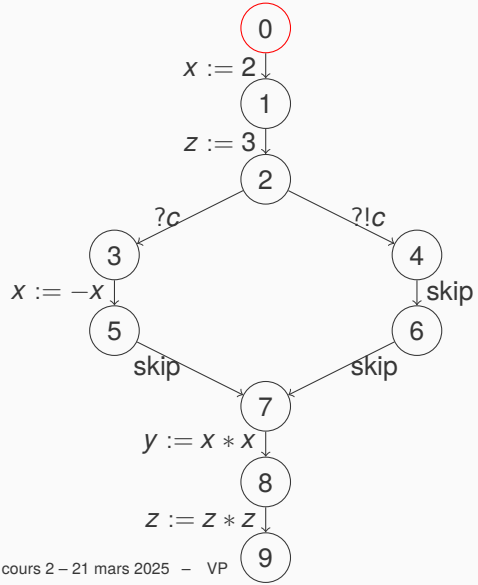
Par contre $\wp(Env_P^\flat)$ ne vérifie pas la condition de chaîne ascendante.

Exemple



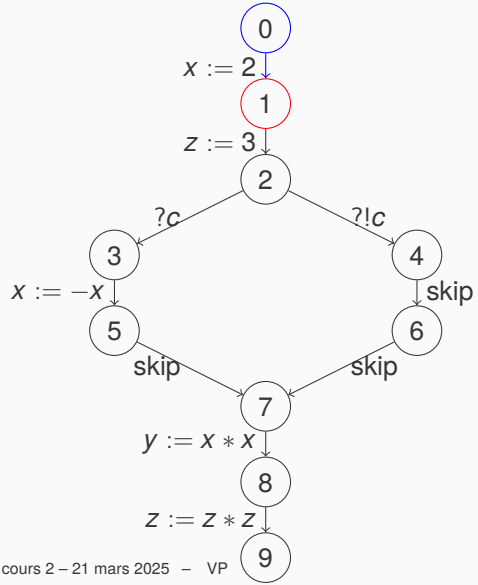
nœuds	environnements
0	
1	
2	
3	
4	
5	
6	

Exemple



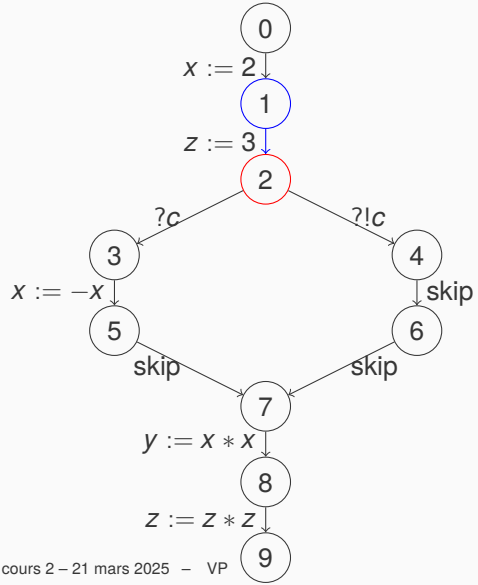
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	
2	
3	
4	
5	
6	

Exemple



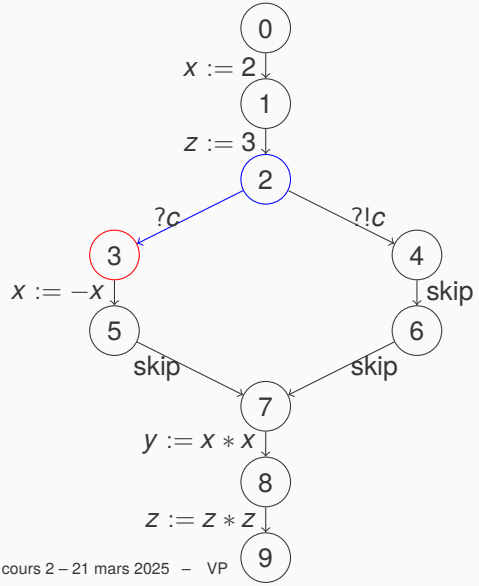
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow n_z; c \leftarrow n_c\}$
2	
3	
4	
5	
6	

Exemple



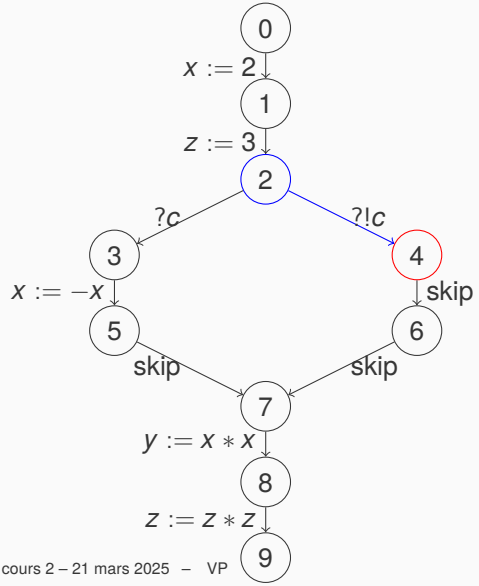
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y; z \leftarrow n_z; c \leftarrow n_c\}$
2	$\{x \leftarrow 2; y \leftarrow n_y; z \leftarrow 3; c \leftarrow n_c\}$
3	
4	
5	
6	

Exemple



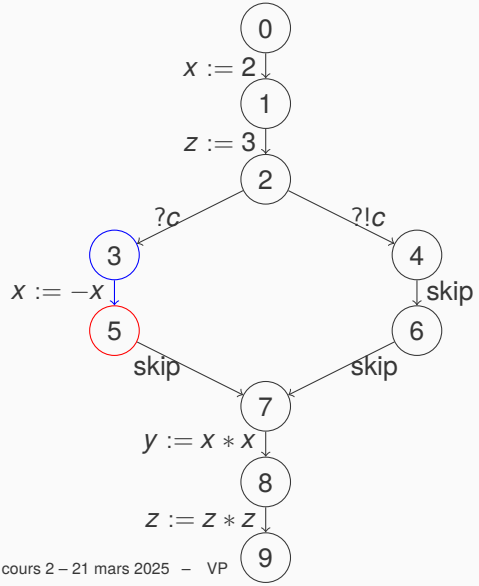
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow n_z; c \leftarrow n_c\}$
2	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}$
3	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
4	
5	
6	

Exemple



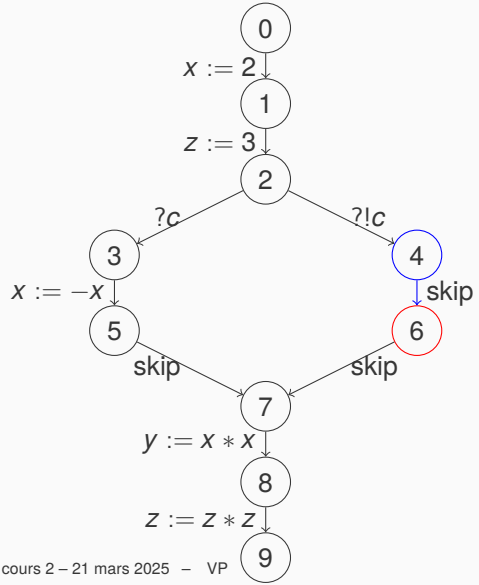
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow n_z; c \leftarrow n_c\}$
2	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}$
3	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
4	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow 0\}$
5	
6	

Exemple



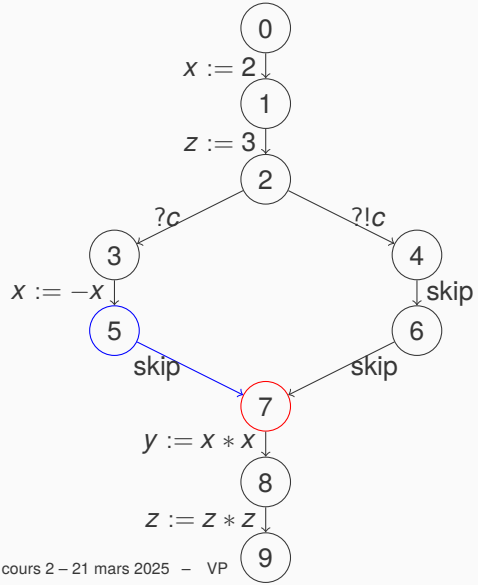
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow n_z; c \leftarrow n_c\}$
2	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}$
3	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
4	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow 0\}$
5	$\{x \leftarrow -2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
6	

Exemple



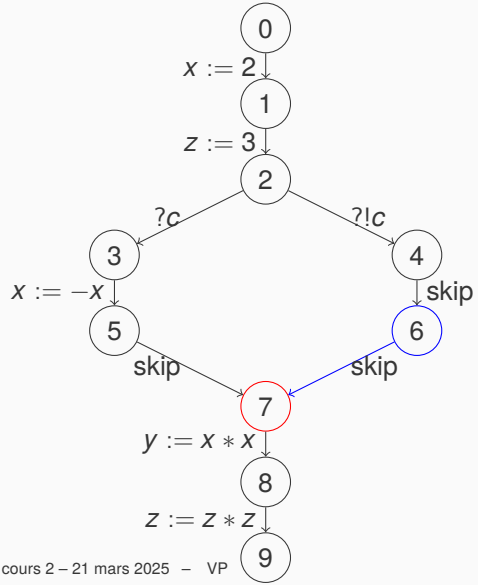
nœuds	environnements
0	$\{\lambda x.n_x \mid n_x \in \mathbb{Z}\}$
1	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow n_z; c \leftarrow n_c\}$
2	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}$
3	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
4	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow 0\}$
5	$\{x \leftarrow -2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$
6	$\{x \leftarrow 2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow 0\}$

Exemple (suite)



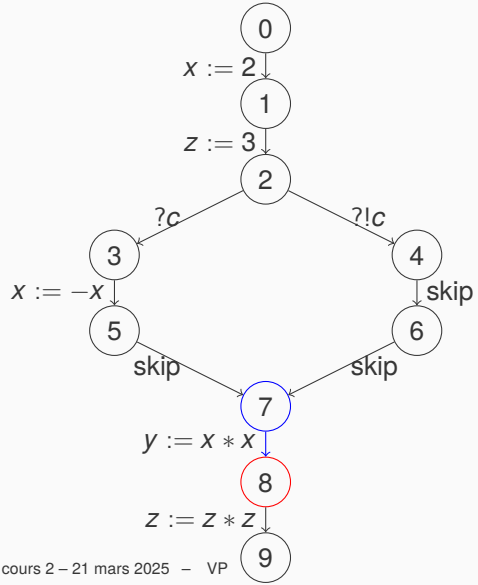
nœuds	environnements
7	
8	
9	

Exemple (suite)



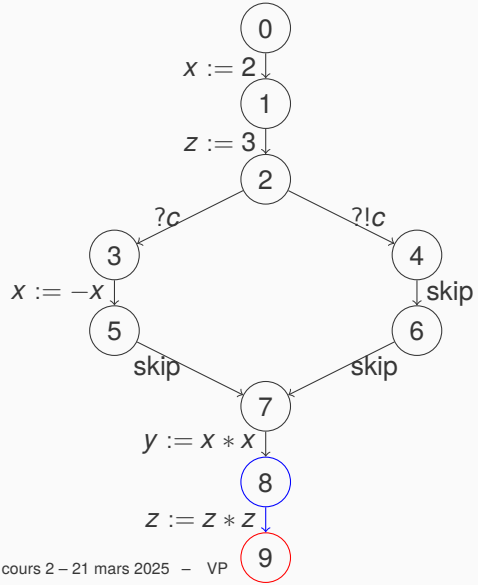
nœuds	environnements
7	$\{x \leftarrow -2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$ $\cup\{x \leftarrow 2; y \leftarrow n_y$ $z \leftarrow 3; c \leftarrow 0\}$
8	
9	

Exemple (suite)



nœuds	environnements
7	$\{x \leftarrow -2; y \leftarrow n_y; z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0 \cup \{x \leftarrow 2; y \leftarrow n_y \; z \leftarrow 3; c \leftarrow 0\}$
8	$\{x \leftarrow -2; y \leftarrow 4; z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0 \cup \{x \leftarrow 2; y \leftarrow 4 \; z \leftarrow 3; c \leftarrow 0\}$
9	

Exemple (suite)



nœuds	environnements
7	$\{x \leftarrow -2; y \leftarrow n_y;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$ $\cup\{x \leftarrow 2; y \leftarrow n_y$ $z \leftarrow 3; c \leftarrow 0\}$
8	$\{x \leftarrow -2; y \leftarrow 4;$ $z \leftarrow 3; c \leftarrow n_c\}; n_c \neq 0$ $\cup\{x \leftarrow 2; y \leftarrow 4$ $z \leftarrow 3; c \leftarrow 0\}$
9	$\{x \leftarrow -2; y \leftarrow 4;$ $z \leftarrow 9; c \leftarrow n_c\}; n_c \neq 0$ $\cup\{x \leftarrow 2; y \leftarrow 4$ $z \leftarrow 9; c \leftarrow 0\}$

Théorème

Tout état associé à un point de programme s dans une trace de P est dans $Coll_P(s)$:

$$\mathcal{C}_P = Coll_P$$

Terminaison

Le calcul de $Coll_P$ par itération ne termine pas. Le prochain cours montrera comment définir des abstractions $\mathcal{A}$ de $Coll_P$

- correctes : telles que $Coll_P(s) \subset \mathcal{A}(s)$
- qui terminent toujours, quel que soit P