Méthodes formelles

Motivations (rappel)

Coût des bugs

- Coûts économique : 64 milliards $/an rien qu'aux US (2002)
- Coûts économique : 150 milliards EUR/an en Europe (2012)
- Coûts humains, environnementaux, etc.

Nécessité d'assurer la qualité des logiciels

Domains critiques

- atteindre le (très haut) niveau de qualité imposée par les lois/normes/assurances/... (ex : DO-178B pour aviation)

Autres domaines

- atteindre le rapport qualité/prix jugé optimal (c.f. attentes du client)

Motivations (2)

Validation et Vérification (V & V)

- Vérification : est-ce que le logiciel fonctionne correctement ?
  - *"are we building the product right ?"*
- Validation : est-ce que le logiciel fait ce que le client veut ?
  - *"are we building the right product ?"*

Quelles méthodes ?

- revues
- simulation/ animation
- tests méthode de loin la plus utilisée
- méthodes formelles encore très confidentielles, mais progresse !

Coût de la V & V

- 10 milliards $/an en tests rien qu'aux US
- plus de 50% du développement d'un logiciel critique (parfois $> 90\%$)
- en moyenne 30% du développement d'un logiciel standard

Un peu d'histoire

"Crise du logiciel" en 1969

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful ! To put it quite bluntly : as long as there were no machines, programming was no problem at all ; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

- Edsger Dijkstra, The Humble Programmer (EWD340)

Système embarqué critique

Système embarqué

Système électronique et informatique autonome, souvent temps-réel, spécialisé dans une tâche bien précise.

Système critique

Système dont une panne peut avoir des conséquences dramatiques.

Exemples de domaine critique

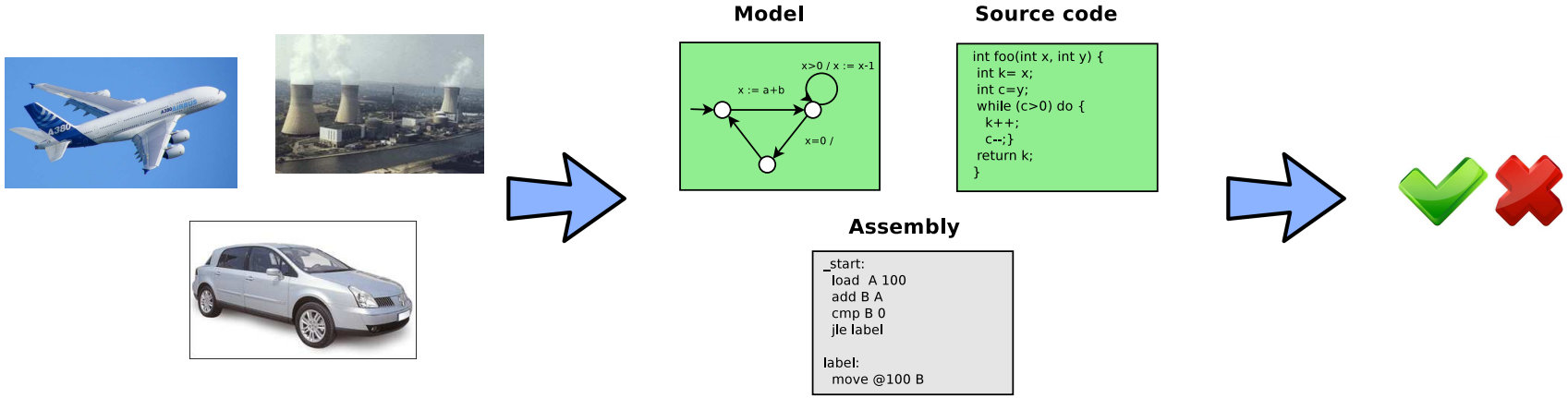
- transport (aérospatial, aéronautique, ferroviaire, automobile)
- énergie, en particulier nucléaire
- gestion des réseaux (telecom, électricité, eau)
- santé
- finance
- applications militaires

Comment assurer le niveau de qualité voulu ?

Testing can only reveal the presence of errors but never their absence.
- E. W. Dijkstra (Notes on Structured Programming, 1972)

Un rêve de logicien

- Between Software Engineering and Theoretical Computer Science
- Goal = proves correctness in a mathematical way



Key concepts : $M \models \varphi$

- M : semantic of the program
- φ : property to be checked
- $\models$: algorithmic check

Kind of properties

- absence of runtime error
- pre/post-conditions
- temporal properties

Eléments clés

Système M

- besoin d'une sémantique : décrire précisément ce qui se passe
- notion d'état mémoire s ($\approx$ associe à chaque variable une valeur)
- définition de la relation de transition $s \xrightarrow{t} s'$: effet de l'instruction t sur l'état s
  - . typiquement, modifie des valeurs associées à des variables dans s
  - . $\{a \mapsto 5, b \mapsto 10\} \xrightarrow{b:=a+b} \{a \mapsto 5, b \mapsto 15\}$
  - . sémantique = interpréteur
- notion d'états accessibles, et de traces

Propriété φ

- implicites (runtime error, mauvais typage, non terminaison)
- ou spécifiable (logique, automates)
- rmq : langage naturel ou semi-formel (UML) trop ambigu
- définition de $\models$: tous les états / traces / ... de M satisfont φ
  - . états$(M) \subseteq$ états(φ), traces$(M) \subseteq$ traces(φ), etc.

Algorithme : attention, indécidable bien souvent. Et cher sinon

Exemple simple

. Imaginons un programme avec seulement des entrées booléennes, et la propriété P : “vérifier que l'on a toujours (si a alors b)”

    . le nombre d'états est fini : on peut tous les énumérer

    . sur chaque état s, on peut facilement vérifier P

. Algorithme de calcul des états ! !

Exemple simple

. Imaginons un programme avec seulement des entrées booléennes, et la propriété P : "vérifier que l'on a toujours (si a alors b)"

. le nombre d'états est fini : on peut tous les énumérer

. sur chaque état s, on peut facilement vérifier P

. Algorithme de calcul des états ! !

Mais en pratique :

- espace des états trop grand ou infini
- propriétés sur un nombre infini de traces potentiellement infinies
- propriétés atomiques complexes (quantificateurs)

Du rêve à la réalité

Industrial reality in some area, especially safety-critical domains (but not only)

- hardware [intel, cadence, etc.], aeronautics [airbus], railroad [metro 14 and many others], smartcards, drivers [SDV, Windows], certified compiler [CompCert], safe DSL for optimizing compilers [alive, in llvm], certified OS [Sel4], memory safety of Linux kernel [with Frama-C], memory safety of parts of Windows [prefast], polar SSL [with Frama-C], many other [Facebook, Amazon, Google, etc.]

Du rêve à la réalité

Industrial reality in some area, especially safety-critical domains (but not only)

- hardware [intel, cadence, etc.], aeronautics [airbus], railroad [metro 14 and many others], smartcards, drivers [SDV, Windows], certified compiler [CompCert], safe DSL for optimizing compilers [alive, in llvm], certified OS [Sel4], memory safety of Linux kernel [with Frama-C], memory safety of parts of Windows [prefast], polar SSL [with Frama-C], many other [Facebook, Amazon, Google, etc.]

Ex : Airbus

Verification of

- runtime errors [Astrée]
- functional correctness [Frama-C]
- numerical precision [Fluctuat]
- source-binary conformance [CompCert]
- ressource usage [Absint]

Du rêve à la réalité

Industrial reality in some area, especially safety-critical domains (but not only)

- hardware [intel, cadence, etc.], aeronautics [airbus], railroad [metro 14 and many others], smartcards, drivers [SDV, Windows], certified compiler [CompCert], safe DSL for optimizing compilers [alive, in llvm], certified OS [Sel4], memory safety of Linux kernel [with Frama-C], memory safety of parts of Windows [prefast], polar SSL [with Frama-C], many other [Facebook, Amazon, Google, etc.]

Ex : Microsoft

Verification of drivers [SDV]

- conformance to MS driver policy
- home developers
- and third-party developers

Du rêve à la réalité

Industrial reality in some area, especially safety-critical domains (but not only)

- hardware [intel, cadence, etc.], aeronautics [airbus], railroad [metro 14 and many others], smartcards, drivers [SDV, Windows], certified compiler [CompCert], safe DSL for optimizing compilers [alive, in llvm], certified OS [Sel4], memory safety of Linux kernel [with Frama-C], memory safety of parts of Windows [prefast], polar SSL [with Frama-C], many other [Facebook, Amazon, Google, etc.]

Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability.

- Bill Gates (2002)

Taxonomie des approaches

Place dans le processus de développement

- a priori
- a posteriori
- approche par raffinement

Objet de la vérification

- modèle vs source (vs binaire)

Niveau d'automatisation

- complètement automatique [enfin, presque]
- manuel
- intermédiaire [annotations]

Propriétés

- prédicats simples ou complexes (logique propositionnelle, 1er ordre QF, 1er ordre, ordres supérieurs)
- aspect temporel (invariance, sûreté, vivacité)
- propriétés implicites ou langage de description

Taxonomie des approches

	system	properties		algorithm		automation
		temporal	atomic pred.	answer	termination	
MC	finite	liveness	boolean	yes/no (+cex)	ok	ok
AI	infinite	invariance *	QF-FOL	yes/ ?	ok	ok
WP	infinite	invariance * termination	FOL	yes/ ?	ok	X

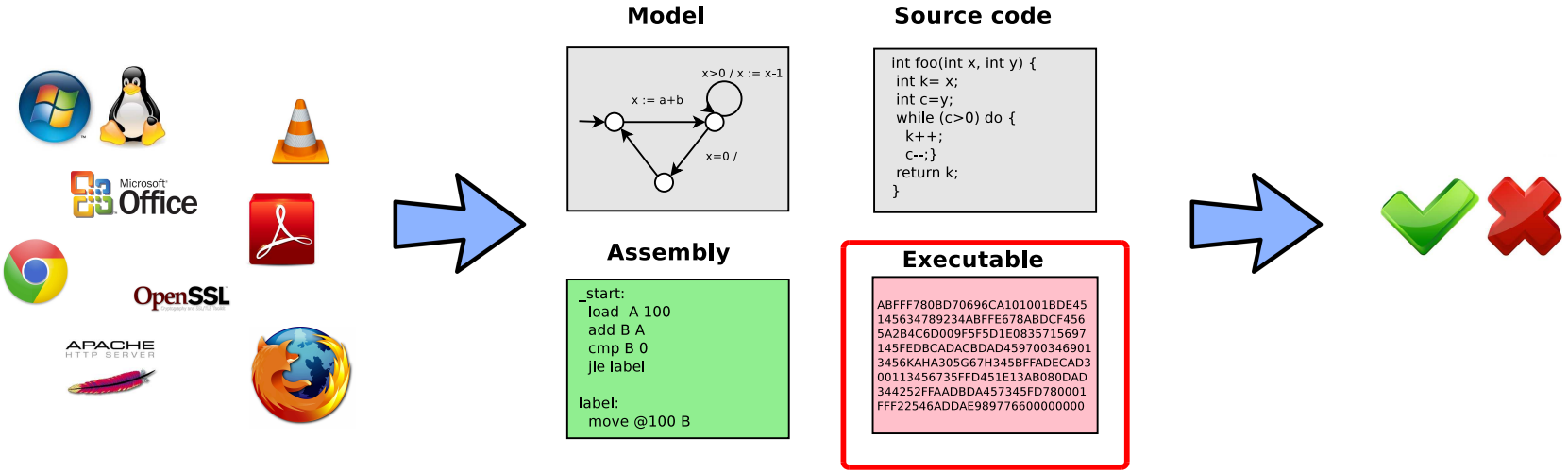
Historique

- 1970's : quelles propriétés, quels modèles ? (sémantique)
- 1980's : premiers algos, classification des problèmes (complexité)
- 1990's : premières études de cas réalistes (protocoles, hardware)
- 2000's : début du software verification

quelques dates clés :
. logique temporelles (Pnueli, 1977),
. lcalcul de WP (Hoare, 197xx),
. linterprétation abstraite (Cousot, 1977),
. lmodel checking (Clarke et al., Sifakis et al., 1981),

Prochain défis

- Apply formal methods to less-critical software
- Very different context : no formal spec, less developer involvement, etc.



Some difficulties

- robustness [w.r.t. software constructs]
- no place for false alarms
- scale
- no model, sometimes no source
- quantitative verification

Guidelines & trends

- find sweetspots [drivers]
- manage abstractions
- reduction to logic

But

- les spécifications permettent de :
  - mieux comprendre le code
  - clarifier les interfaces (modules/fonctions)
  - faciliter la maintenance
  - tester le logiciel en boite noire
- les spécifications formelles permettent de :
  - raisonner formellement sur le programme
  - vérifier que le code correspondant les satisfait
  - utiliser des outils

Éléments clés

Langages de contrat :

- précondition, postcondition [appelant, appelé]
- frame / assigns
- prédicats avancés : built-in (\valid), $\forall$, $\exists$, etc.
- boucles : invariants et variants
- assert intermédiaires [hints]

Aspects temporels : cf model checking

Bibliographie : langages de spécification

- Eiffel : Analysis, Design and Programming Language 2nd edition. 2006.
  http://www.ecma-international.org/publications/standarc

- Java Modeling Language (JML). 2000.
  http://www.eecs.ucf.edu/~leavens/JML/index.shtml
- Spec♯. 2004.
  http://research.microsoft.com/en-us/projects/specsharp

- ANSI C Specification Language (ACSL). 2008.
  http://frama-c.cea.fr/acsl.html
- Executable ANSI C Specification Language (E-ACSL). 2012.
  http://frama-c.cea.fr/eacsl.html
- Spark-2014. 2013.
  http://www.spark-2014.org

Triplet de Hoare

- Un triplet de Hoare est un triplet noté $\{P\}i\{Q\}$ où P et Q sont des propriétés (formules logiques) et i est une instruction du programme.
- P est appelé la pré-condition de i
- Q est appelé la post-condition de i
- Informellement, un triplet de Hoare est valide si la propriété suivante est vraie :

  si la pré-condition P est valide avant d'exécuter l'instruction i, alors la post-condition Q sera valide après l'exécution de i

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{faux\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Exemples

Les triplets de Hoare suivants sont-ils valides ?

- $\{x = 10\}x := x+1\{x = 11\}$? oui
- $\{c = 0\}$if c then $x = c$ else $x = 1$ fi$\{x = 0 \wedge c = 0\}$? non
- $\{c = 2\}$if c then $x = c$ else $x = 1$ fi$\{x = 2 \wedge c = 2\}$? oui
- $\{\text{faux}\}x = 0; y = 1\{x = 0 \wedge y = 2\}$? oui
- $\{i = 1\}$while i <= 10 do $i := i + 1$ done$\{i = 10\}$? non

Comment effectuer une preuve ?

Prouver ? Oui, mais comment ?

- prouver à la main (comme n'importe quel théorème de maths)
  - impossible en pratique
- prouver à l'aide d'un assistant de preuves : outil pour assister un humain dans l'activité de preuves en le guidant, en résolvant les parties triviales ou calculatoires, et en vérifiant leur validité
  - Coq
  - Isabelle
  - PVS
- prouver avec des prouveurs automatiques : outil "presse-bouton" effectuant les preuves de manière automatique
  - Alt-Ergo
  - Simplify
  - Z3

Gestion des obligations de preuve

Contexte de la preuve de programmes (de taille réelle)

- les obligations de preuve sont générées automatiquement par des outils
- plusieurs milliers d'obligations de preuve (voire plus)
- plusieurs centaines d'hypothèses par obligation de preuve (voire plus)

Faisabilité des preuves

- infaisable par un humain
- reste très lourd avec un assistant de preuves
- emploi de prouveurs automatiques indispensables
- les prouveurs automatiques ne savent pas tout faire

Analyse flot de données (data flow)

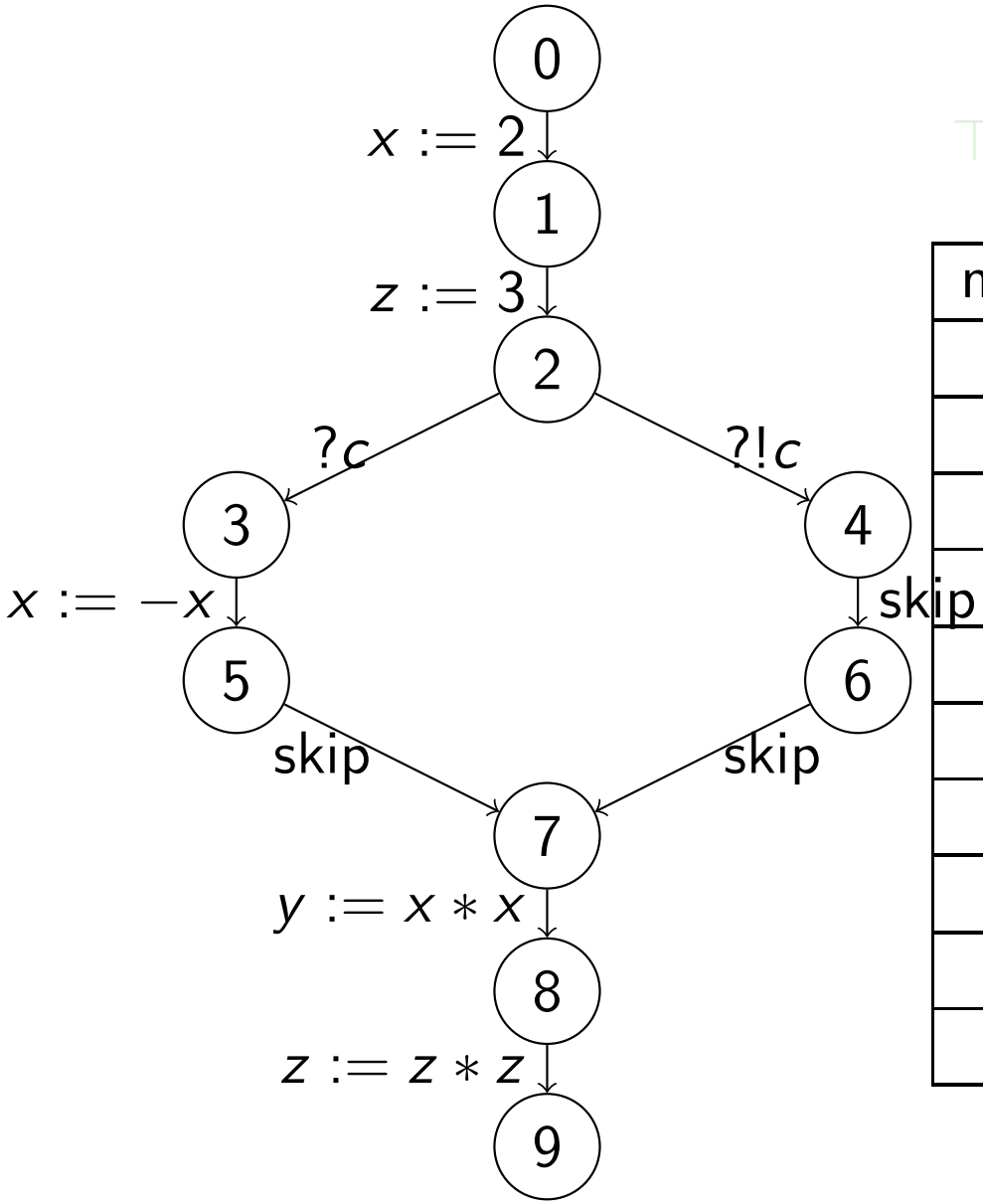
Vue informelle

- 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 ?

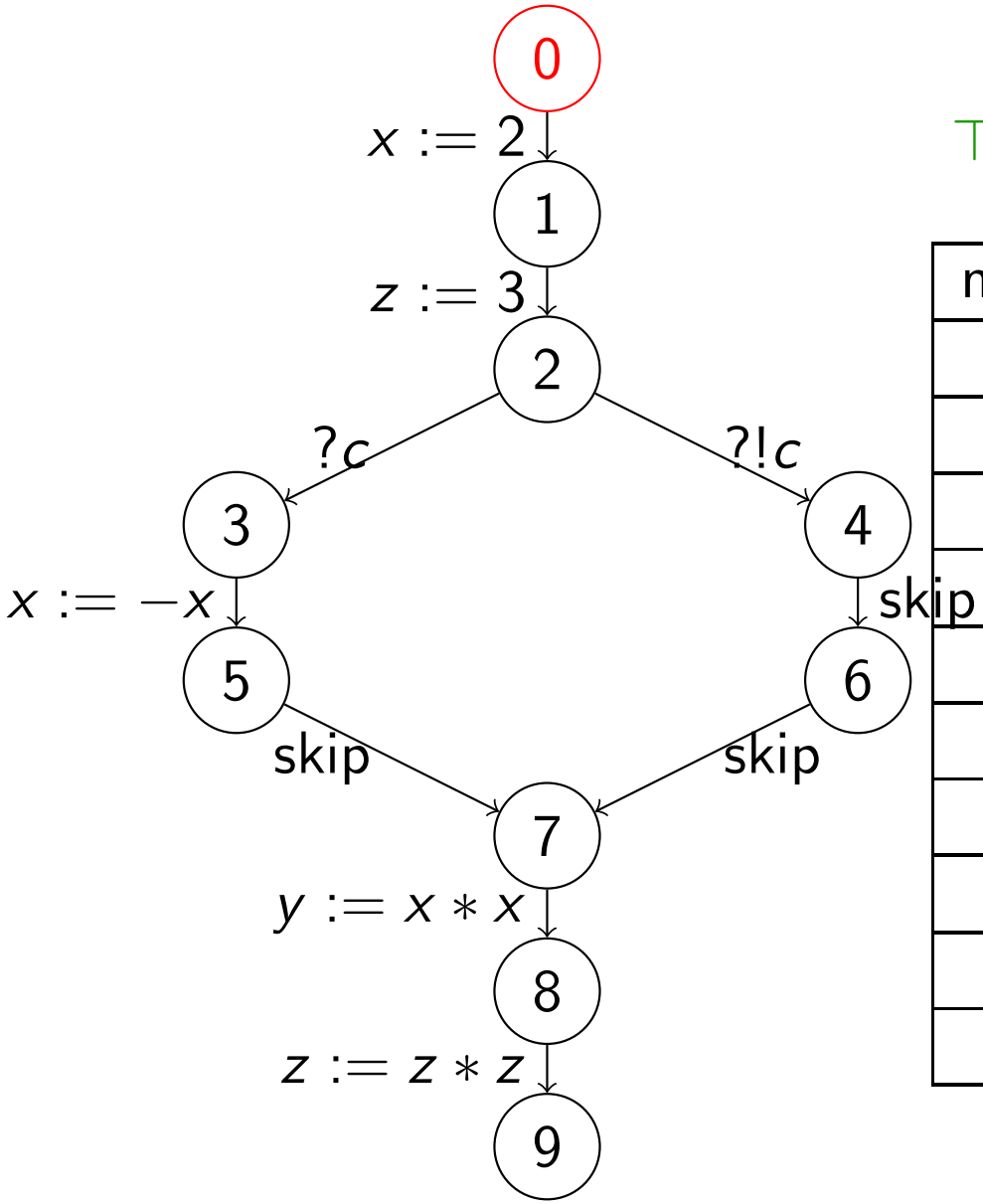
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

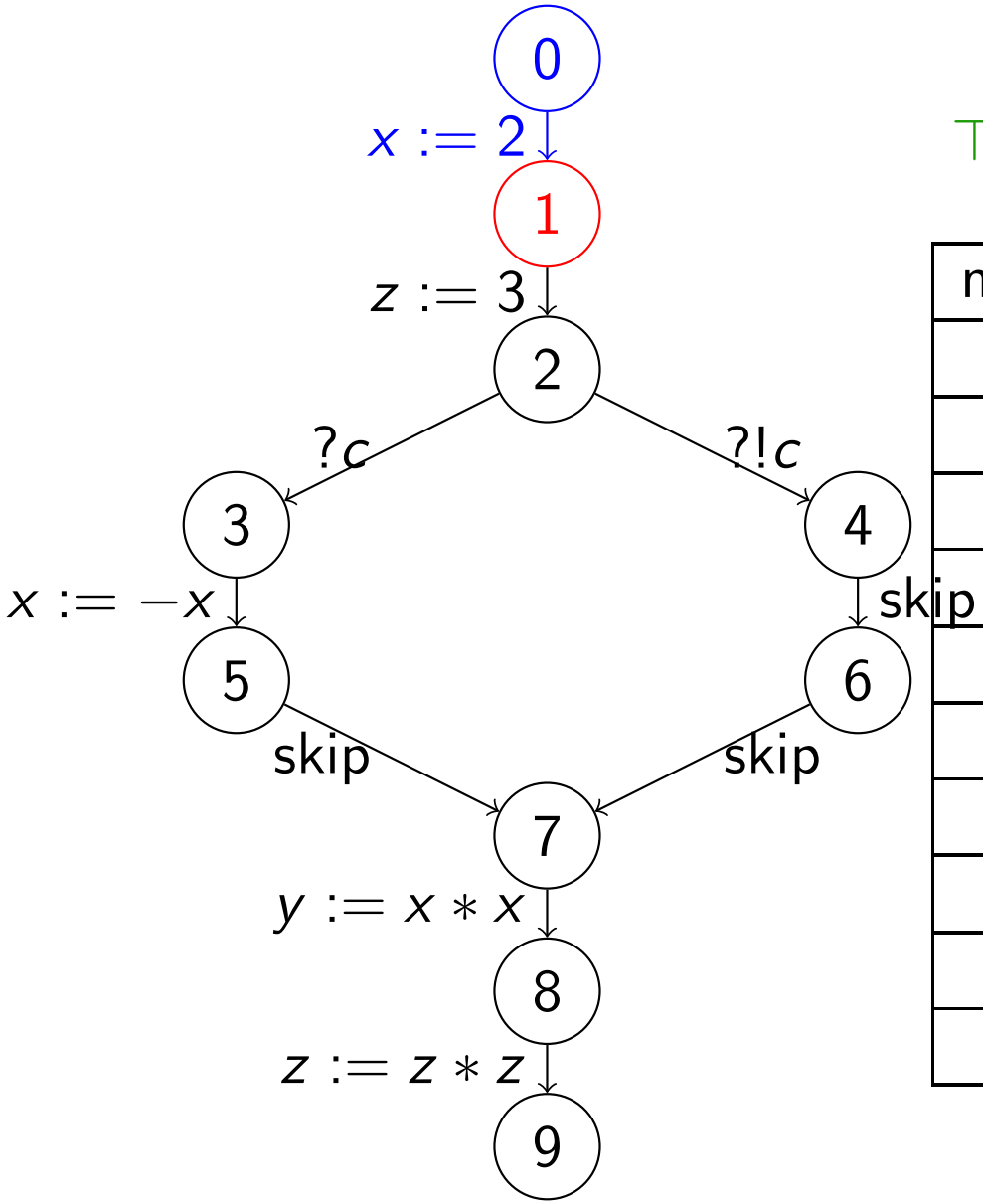
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

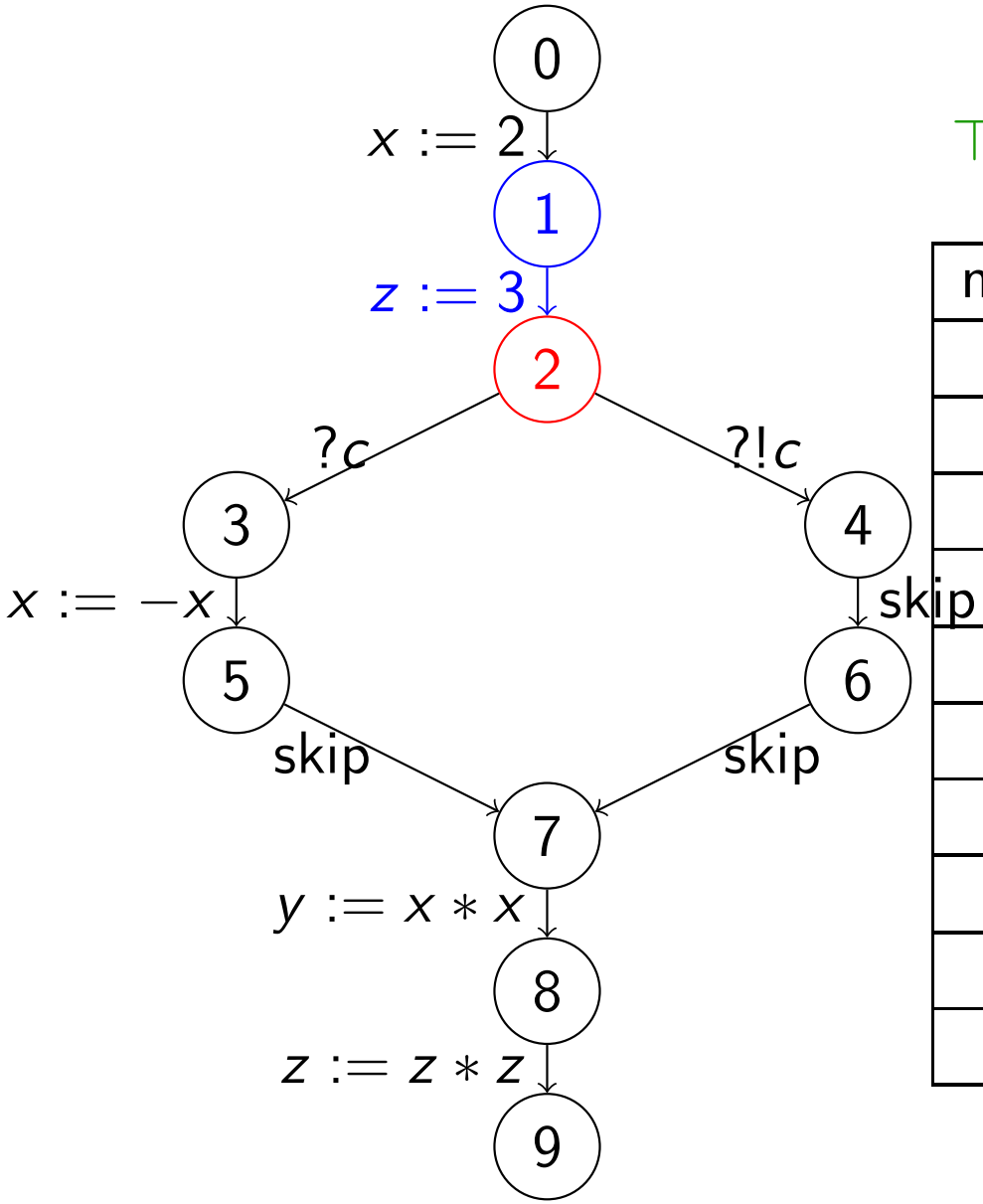
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	∓ 2	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

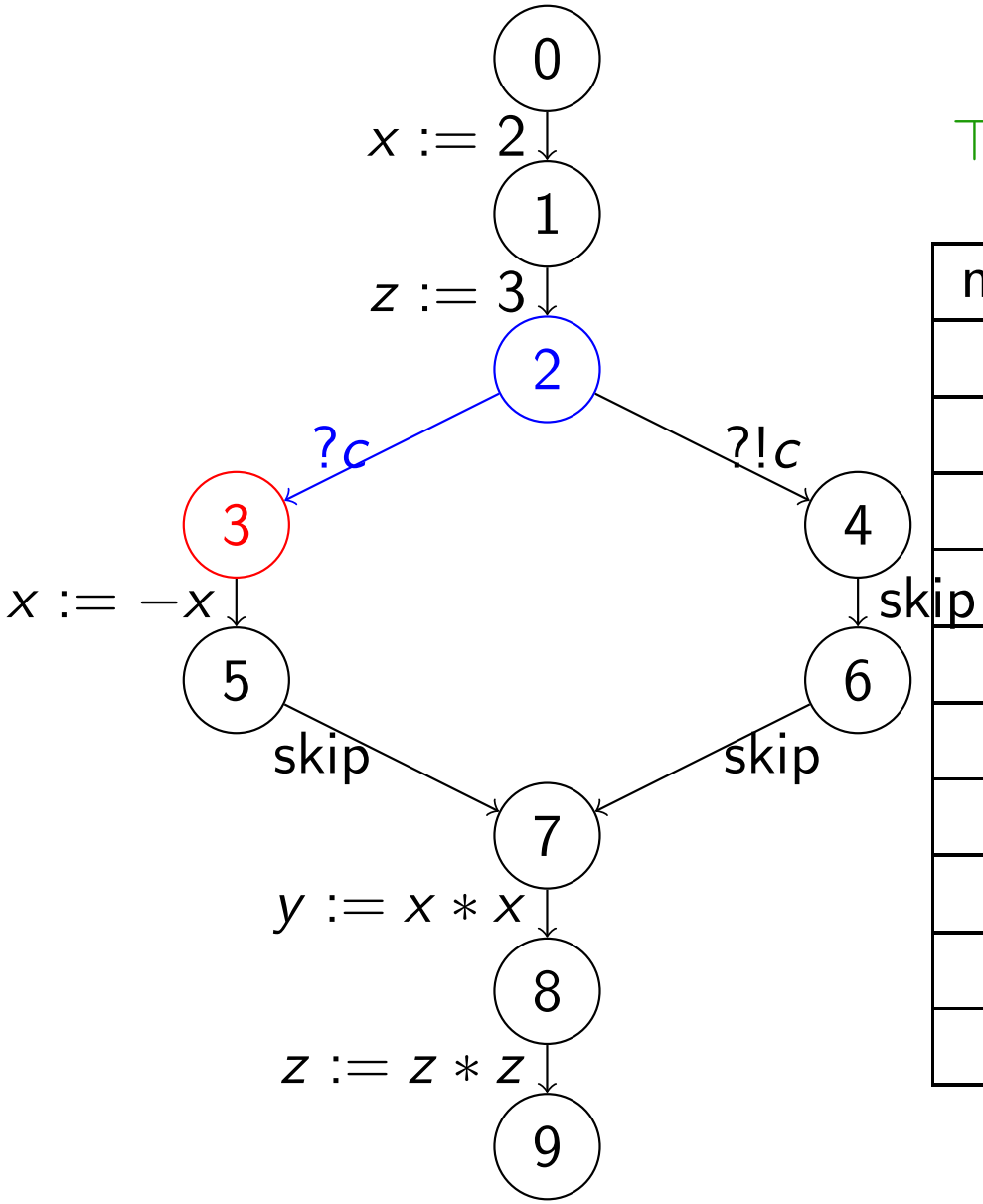
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

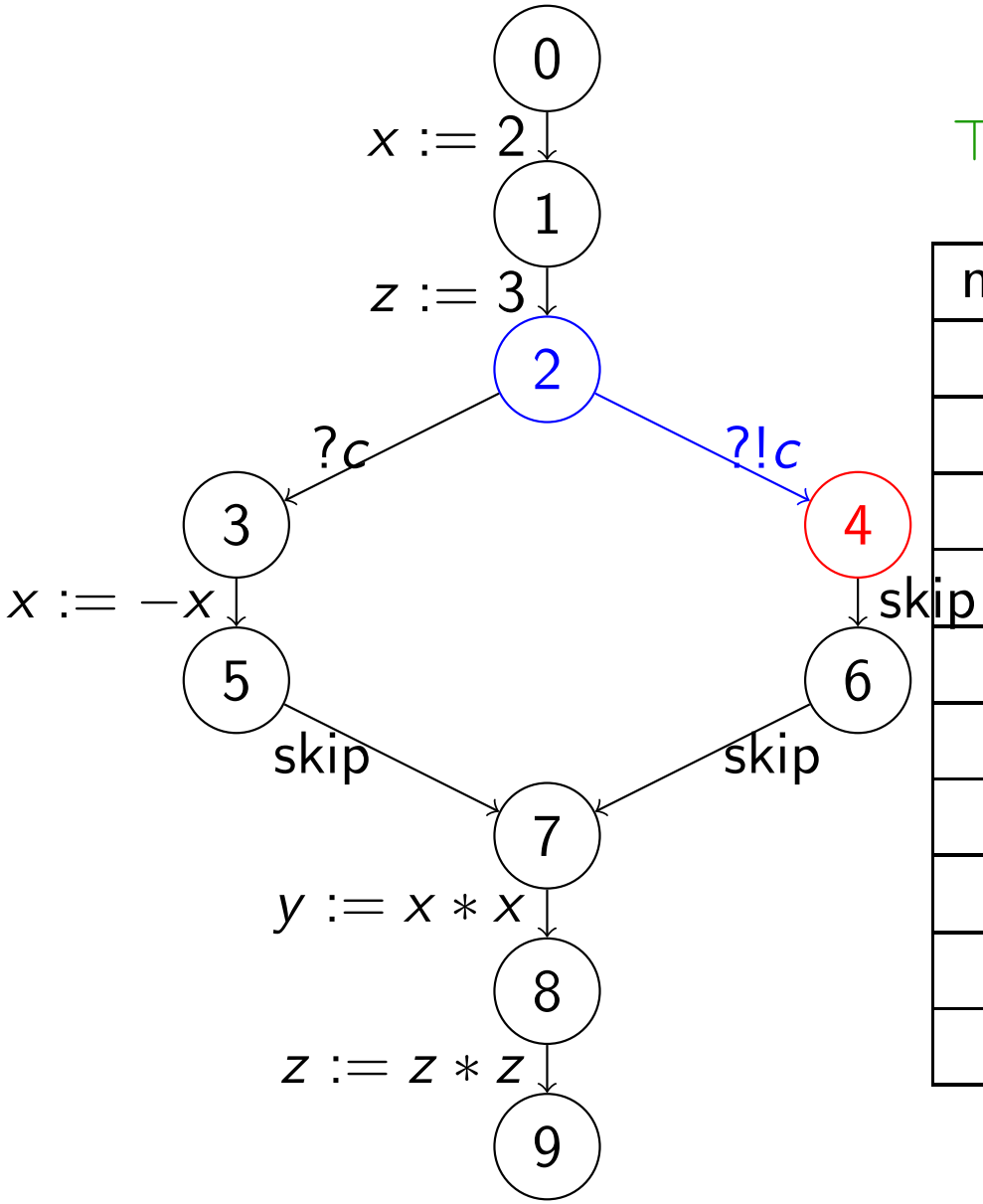
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

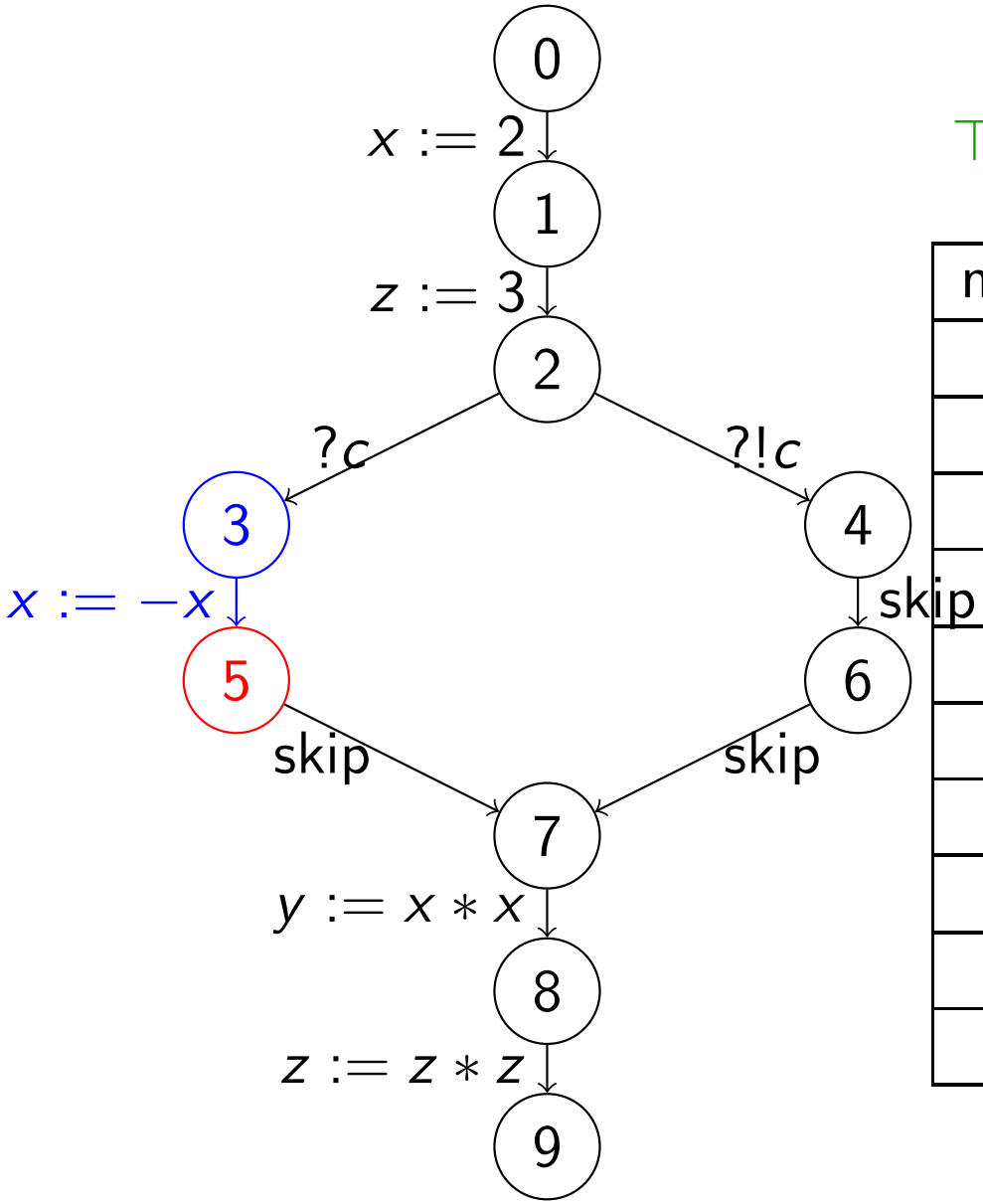
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

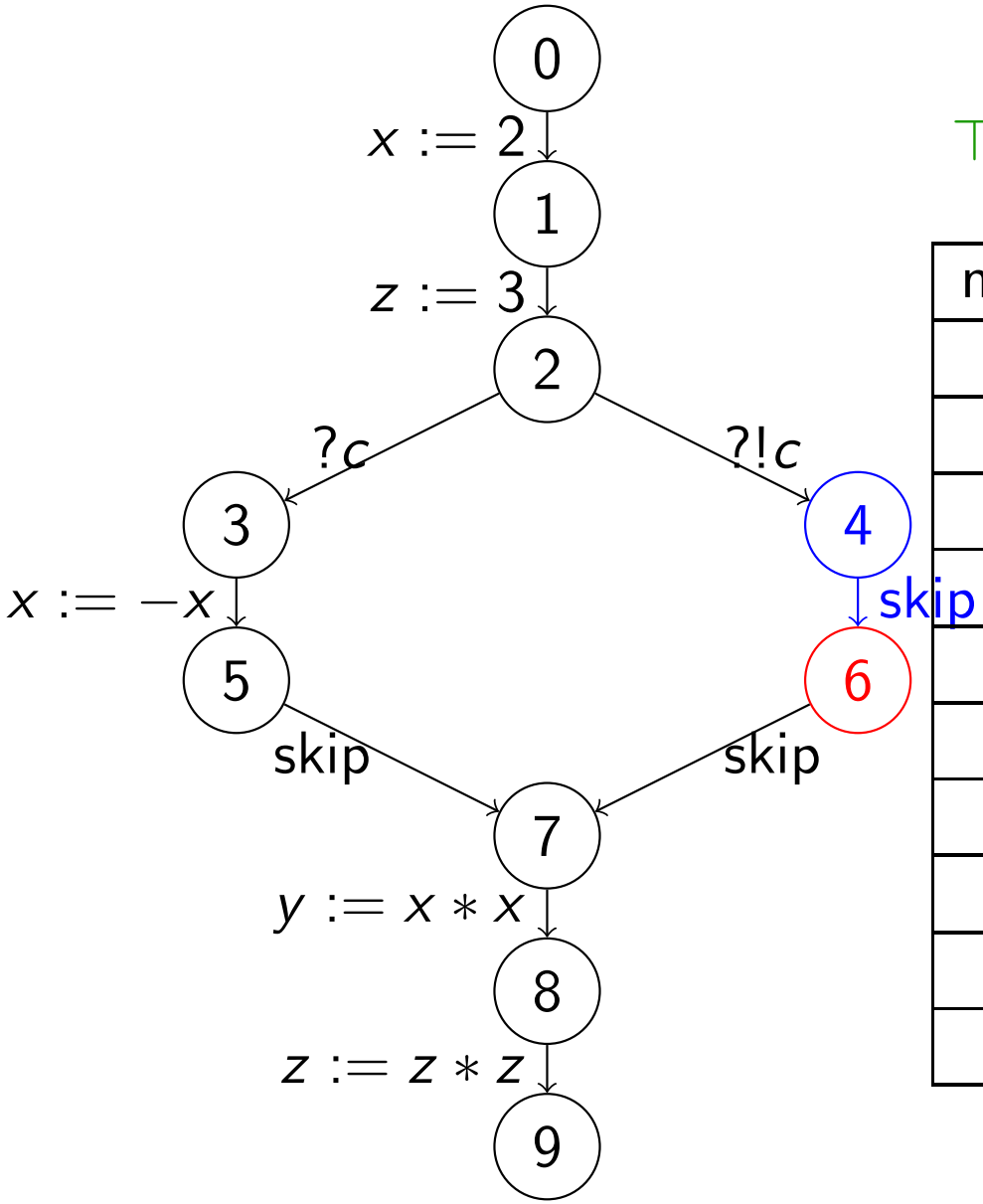
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

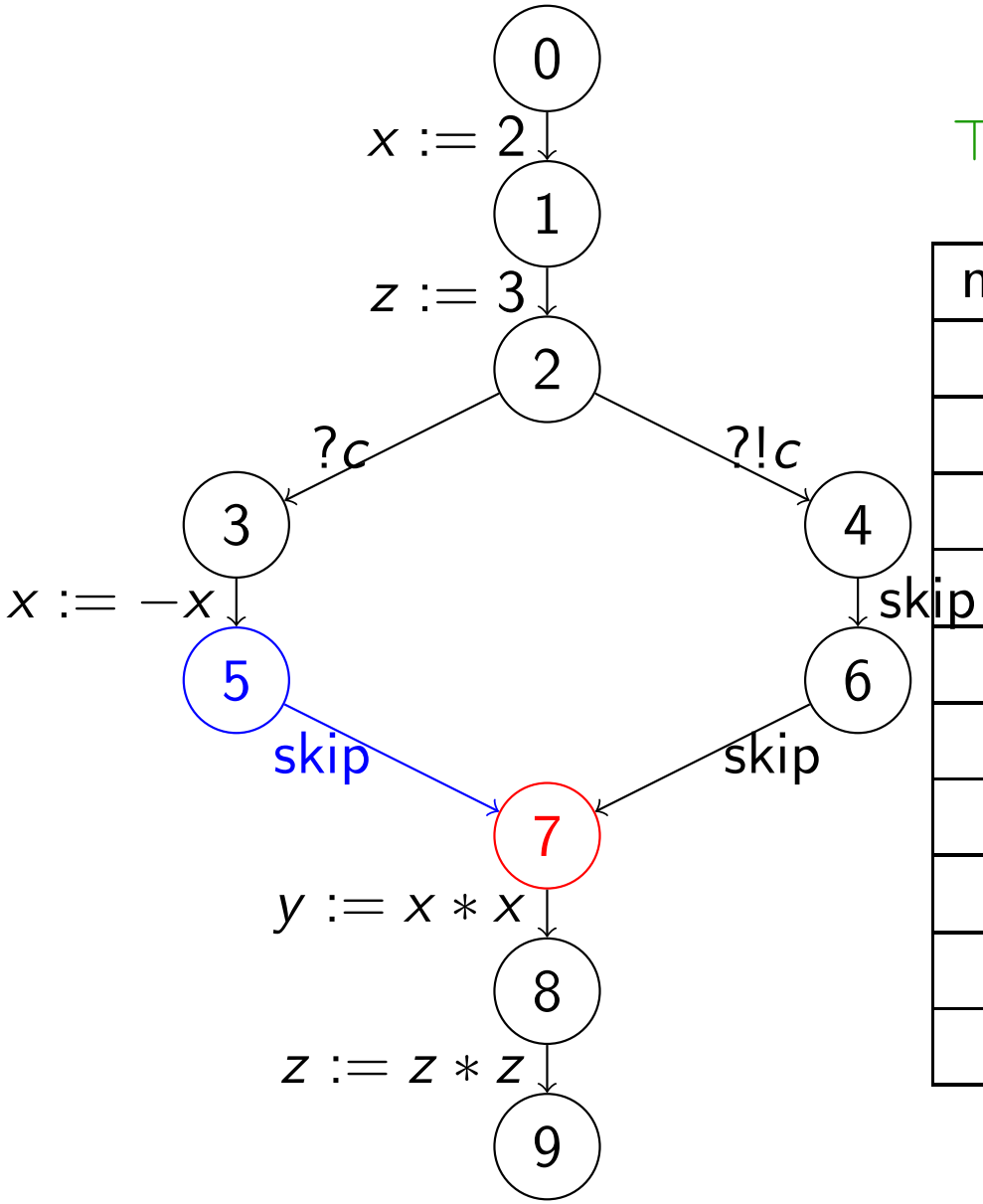
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

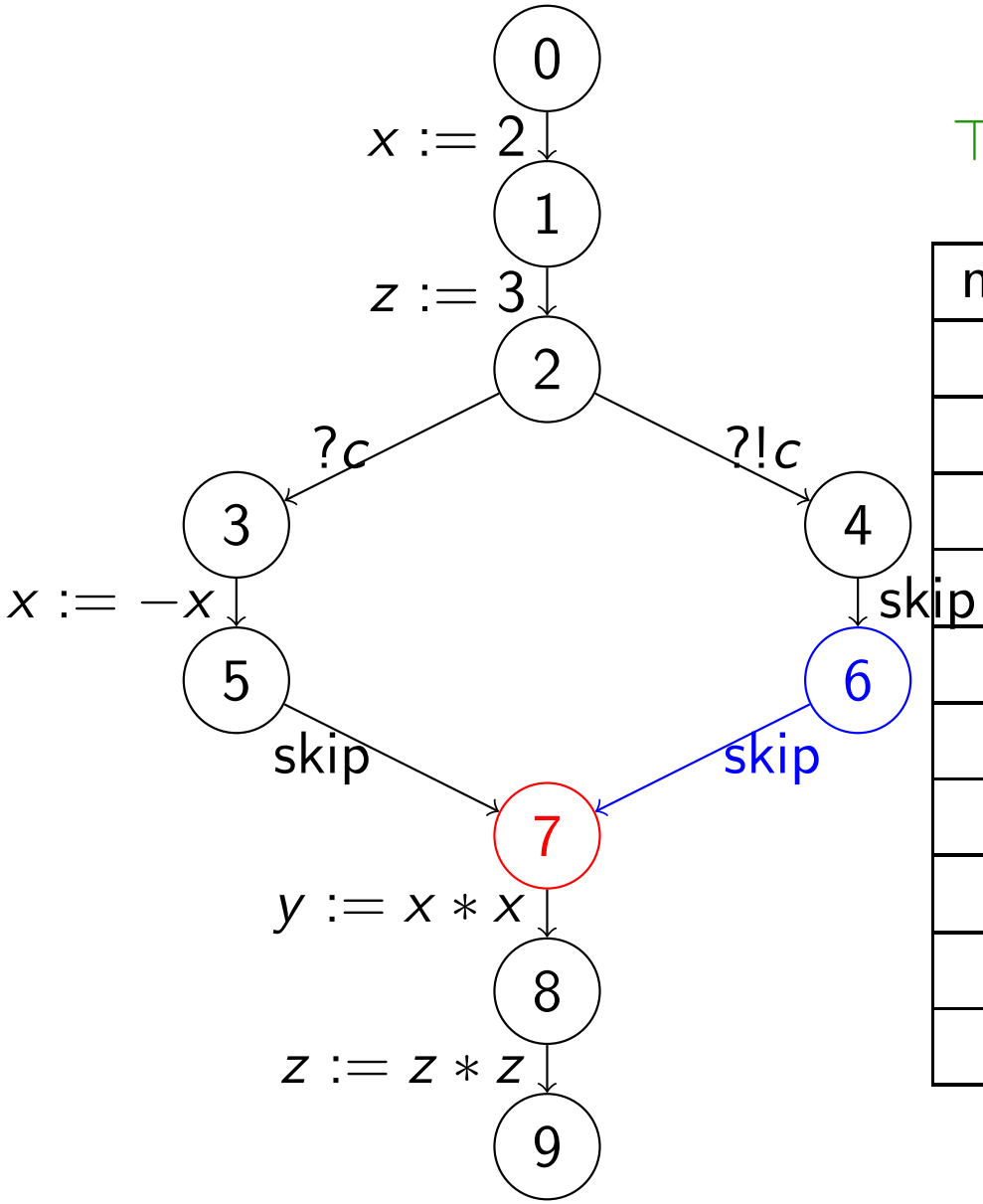
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	-2	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

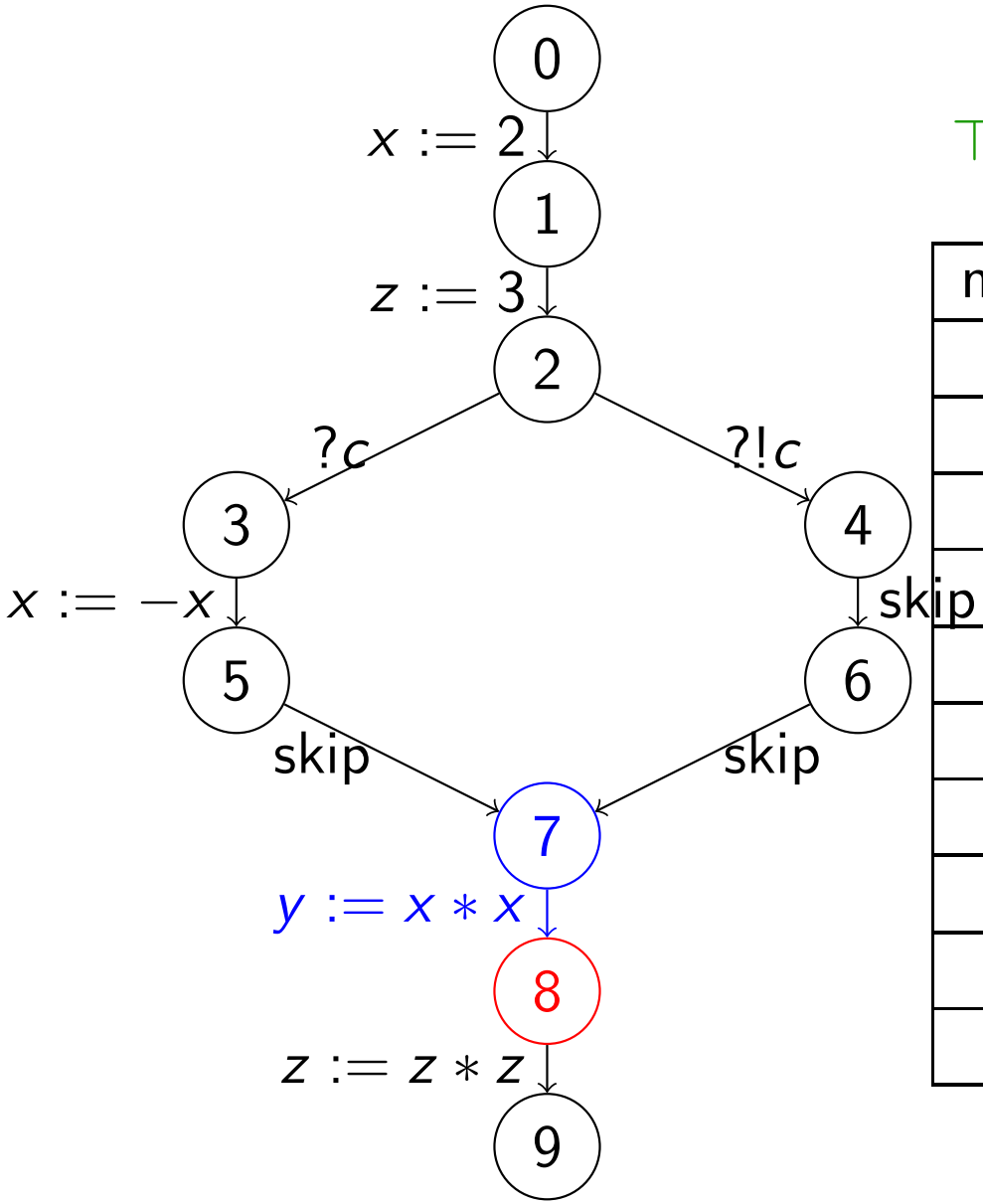
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

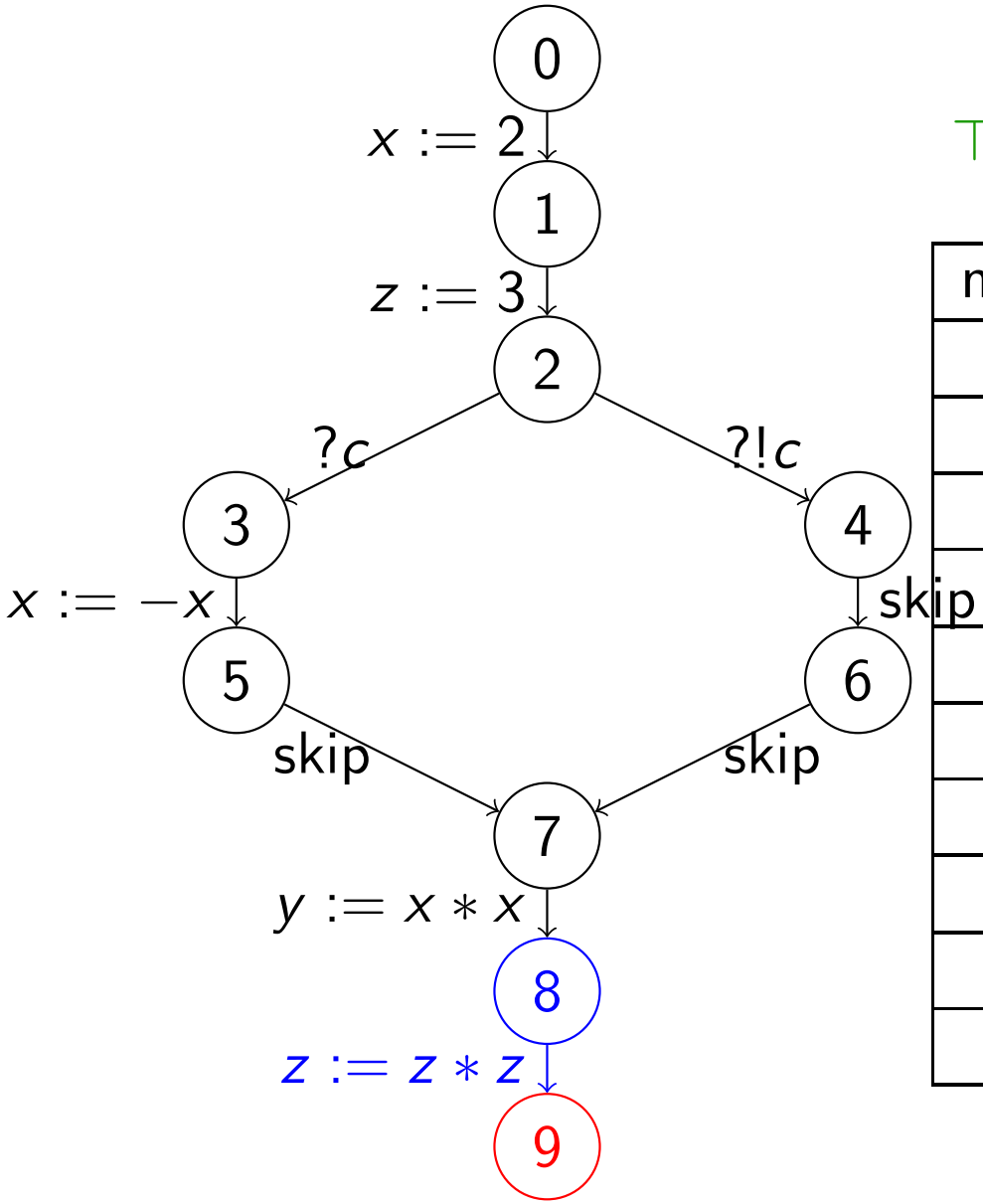
Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

Exemple de propagation



$\top$ = "pas d'information"

nœud	c	x	y	z
0	$\top$	$\top$	$\top$	$\top$
1	$\top$	2	$\top$	$\top$
2	$\top$	2	$\top$	3
3	$\top$	2	$\top$	3
4	0	2	$\top$	3
5	$\top$	-2	$\top$	3
6	0	2	$\top$	3
7	$\top$	$\top 2$	$\top$	3
8	$\top$	$\top$	$\top$	3
9	$\top$	$\top$	$\top$	9

Pour aller plus loin

- correct si les opérations abstraites sont surapprochées
- interprétation abstraite : une connexion de Galois assure l'existence d'une meilleure approximation
- termine si le treillis n'a pas de chaine ascendante infinie
- widening pour assurer la terminaison
- industrie : ASTREE, SDV, Clousot, Absint, Polyspace, Frama-C, Fluctuat, etc.