DIM0436

30. Intepretação abstrata 1

2014115

Sumário

Definições

Definição

A **interpretação abstrata** (IA) é uma teoria de aproximação **correta** da **semântica**.

Definição (Corretude)

Uma aproximação (abstração) é dita correta se o conjunto dos comportamentos do programa fonte original é incluído no conjunto dos comportamentos da abstração.

A base

Princípio de base

- **Aproximar a semântica concreta** por uma semântica abstrata
- **Perda de informação**
- **Objetivo:** achar a melhor aproximação possível, em relação ao objetivo determinado e às restrições existentes (essas aulas não tratam disso)

Fundamentos

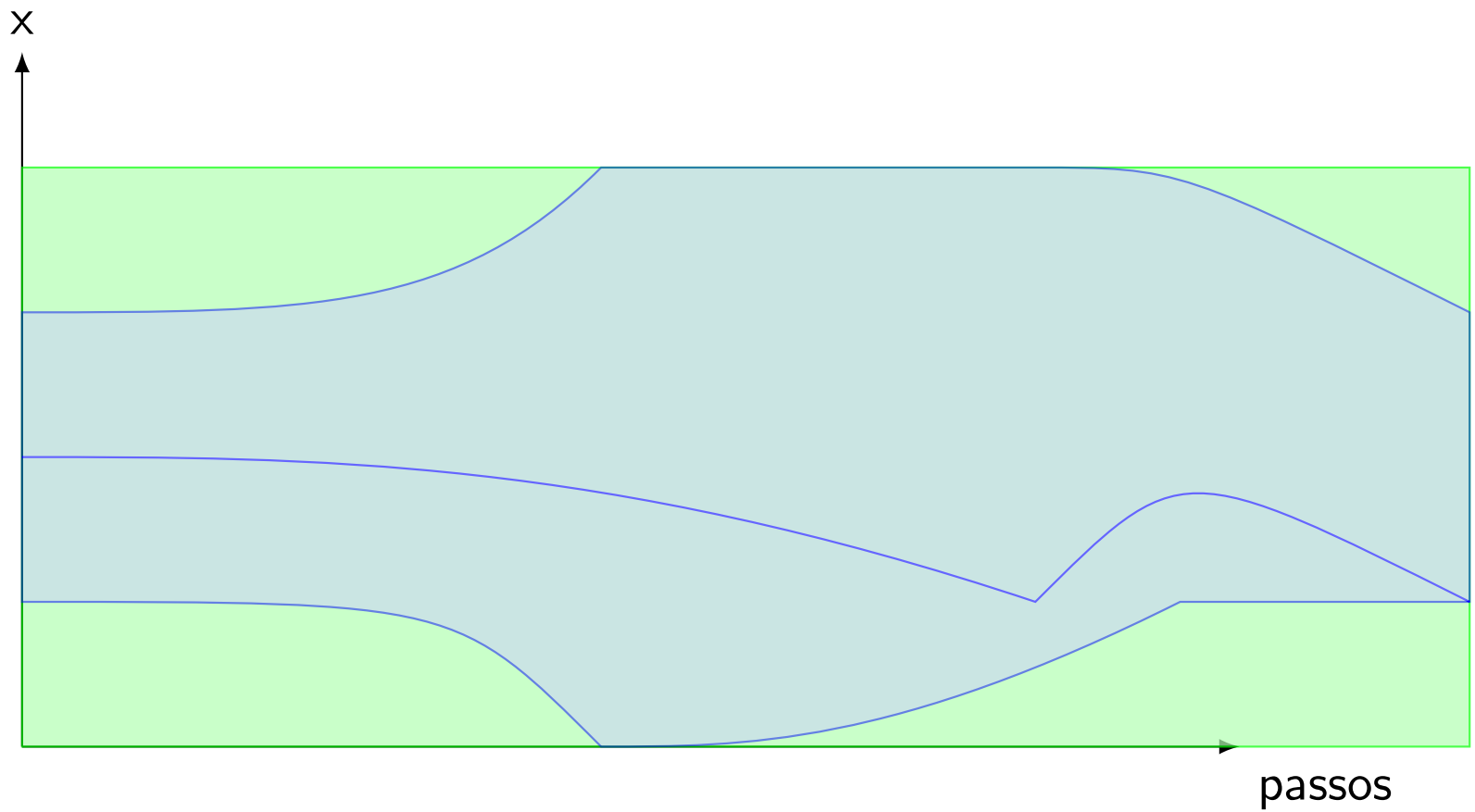
Fundamentos teóricos

- **Operadores** para construir uma semântica abstrata
- Condições sobre as relações entre domínios abstrato e concreto
- Garantia de **corretude**
- Garantia de **completude**
- Artigo fundamental [CC77]

Corretude

Definição (Corretude)

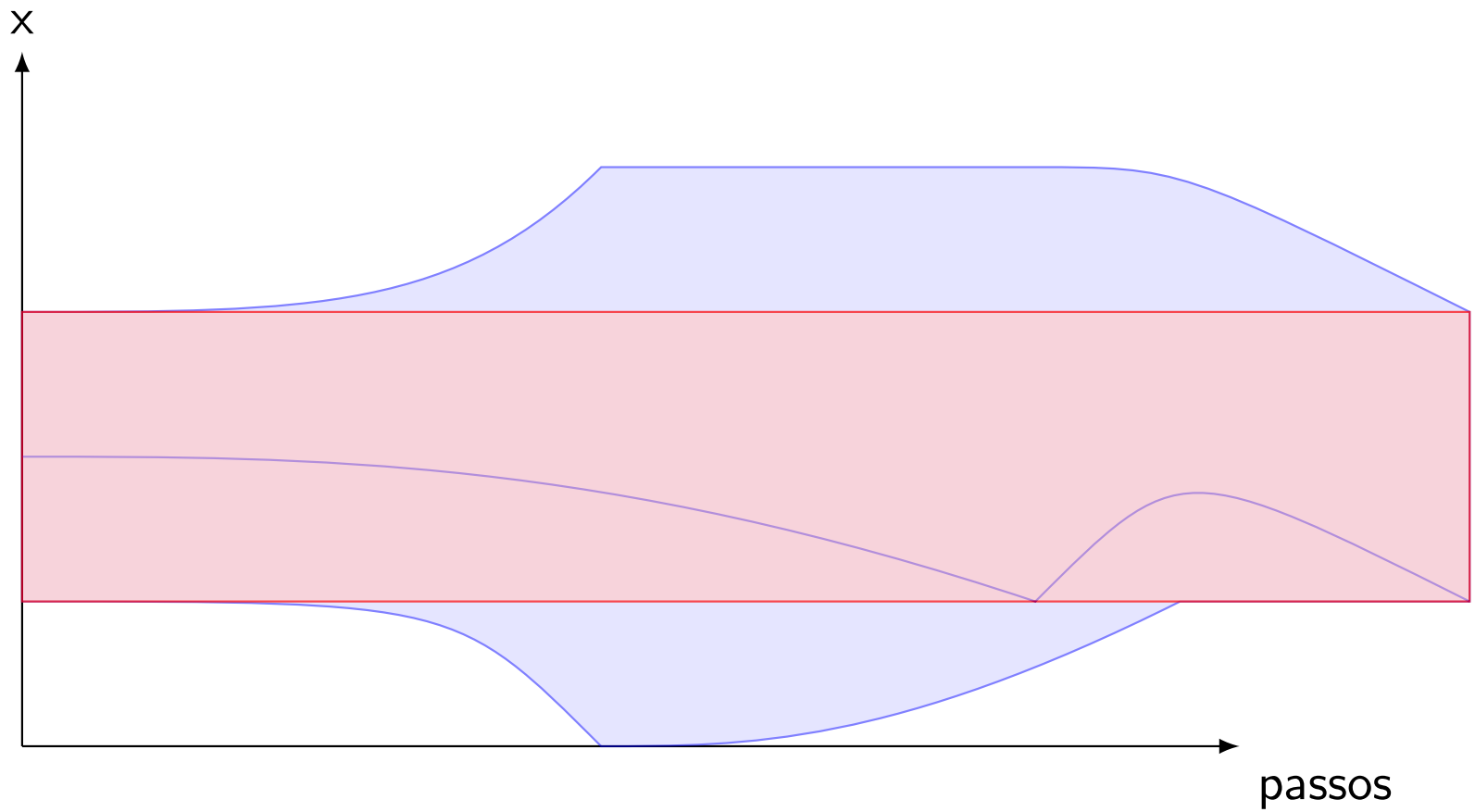
Todos os comportamentos da semântica concreta são preservados pela aproximação.



Completude

Definição (Completude)

Todo comportamento da aproximação corresponde a um comportamento da semântica concreta.



Corretude e completude

Tipos de analisadores estáticos

- incorreto e incompleto ...

Corretude e completude

Tipos de analisadores estáticos

- incorreto e incompleto ...
- **incorreto e completo**: é o princípio dos testes

Corretude e completude

Tipos de analisadores estáticos

- incorreto e incompleto ...
- **incorreto e completo**: é o princípio dos testes
- **correto e incompleto**: é o princípio dos interpretadores abstratos

Corretude e completude

Tipos de analisadores estáticos

- incorreto e incompleto ...
- **incorreto e completo**: é o princípio dos testes
- **correto e incompleto**: é o princípio dos interpretadores abstratos
- **correto e completo**: é impossível automaticamente (teorema de Rice)

Observação

Aqui as noções de corretude e de completude são **invertidas** em relação ao vocabulário dos **testes**.

Exemplo não computacional

Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:

Exemplo não computacional

Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)

Exemplo não computacional

Consideramos os estudantes presentes nessa sala . . .

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome

Exemplo não computacional

Consideramos os estudantes presentes nessa sala . . .

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)

Exemplo não computacional

Consideramos os estudantes presentes nessa sala . . .

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)
    - ⋆ é suficiente para a ausência (é funciona relativamente bem para a presença)

Exemplo não computacional

Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)
    - ⋆ é suficiente para a ausência (é funciona relativamente bem para a presença)
- Para responder à pergunta "Tinha alguem com n anos de idade na sala?", pode-se:

Exemplo não computacional

Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)
    - ⋆ é suficiente para a ausência (é funciona relativamente bem para a presença)
- Para responder à pergunta "Tinha alguem com n anos de idade na sala?", pode-se:
  - conservar só o idade dos estudantes (nomes são supérfluos)

Exemplo não computacional

Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)
    - ⋆ é suficiente para a ausência (é funciona relativamente bem para a presença)
- Para responder à pergunta "Tinha alguem com n anos de idade na sala?", pode-se:
  - conservar só o idade dos estudantes (nomes são supérfluos)
  - conservar só o mínimo m e o máximo M

Exemplo não computacional

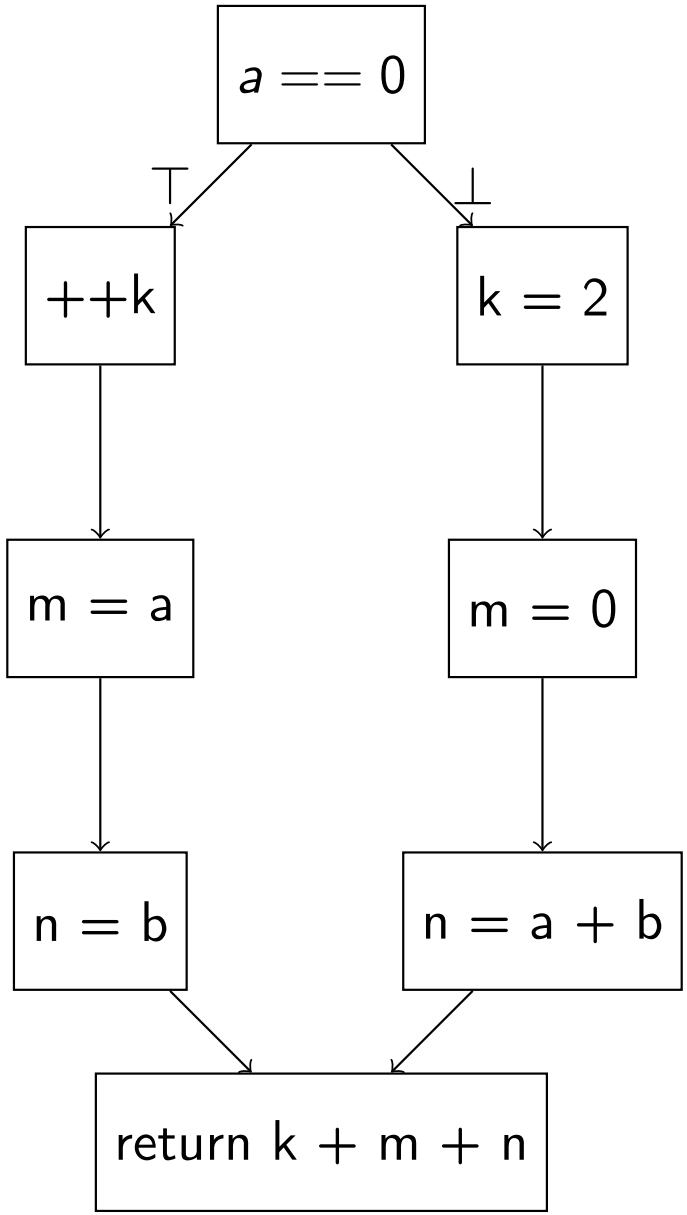
Consideramos os estudantes presentes nessa sala ...

- Para determinar a ausência de certas pessoas, pode-se:
  - ter a lista con os nomes e a matrícula (única)
  - usar só o nome
    - ⋆ é mais impreciso (a presênça não pode ser **demostrada** por causa de homônimos)
    - ⋆ é suficiente para a ausência (é funciona relativamente bem para a presença)
- Para responder à pergunta "Tinha alguem com n anos de idade na sala?", pode-se:
  - conservar só o idade dos estudantes (nomes são supérfluos)
  - conservar só o mínimo m e o máximo M
    - ⋆ se uma pessoa tiver mais de M anos, temos a certeza da sua ausência.

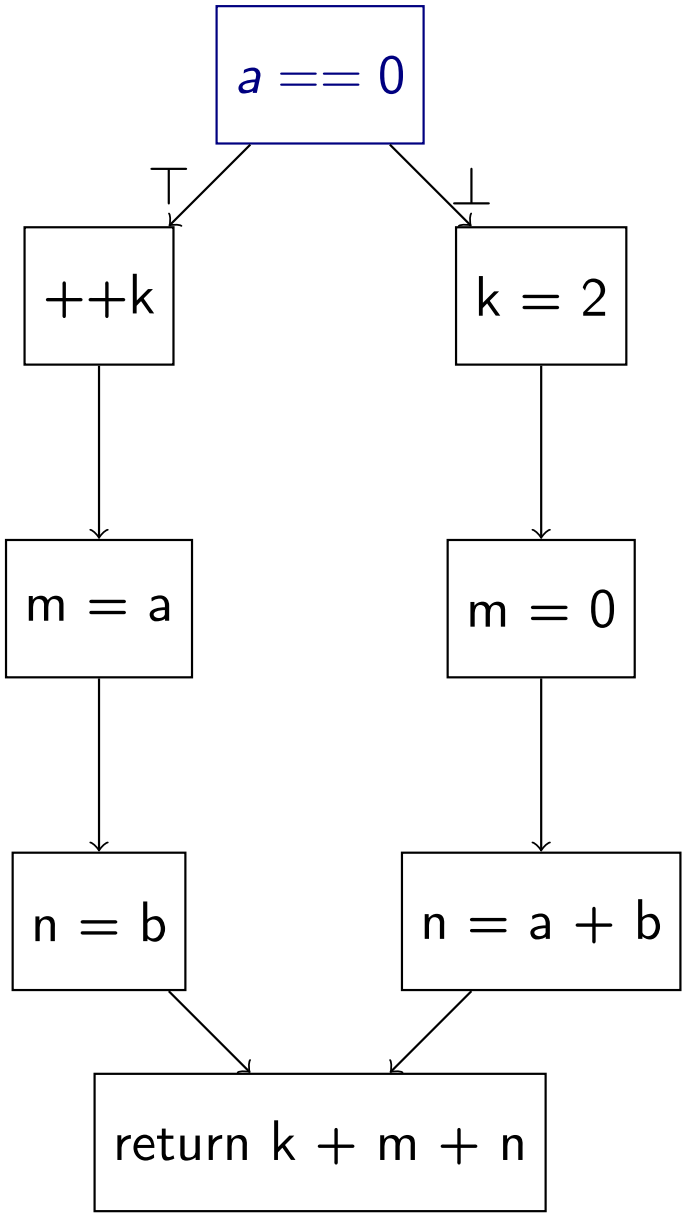
Um exemplo computacional

```
int foo(int a, int b) {
  int k = 1;
  int m, n;
  if (a == 0) {
    ++k;
    m = a;
    n = b;
  } else {
    k = 2;
    m = 0;
    n = a + b;
  }
  return k + m + n;
}
```

CFG



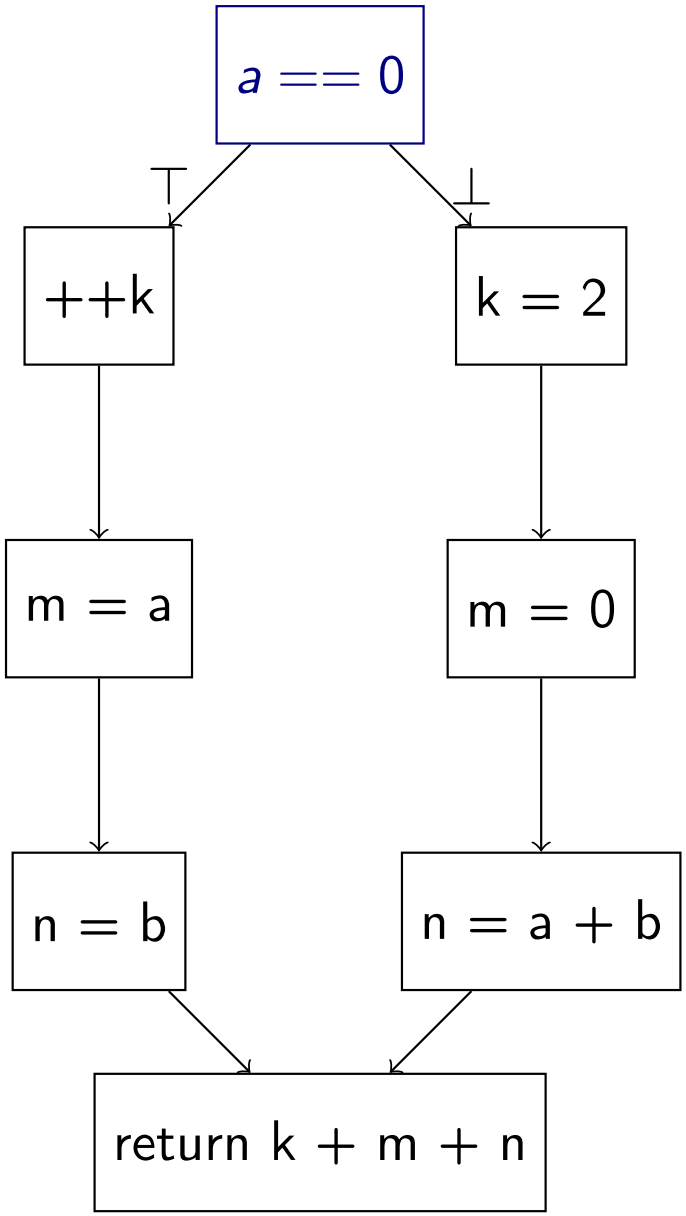
Execução concreta



Parâmetros	
a	0
b	7

Variáveis
k
m
n
return

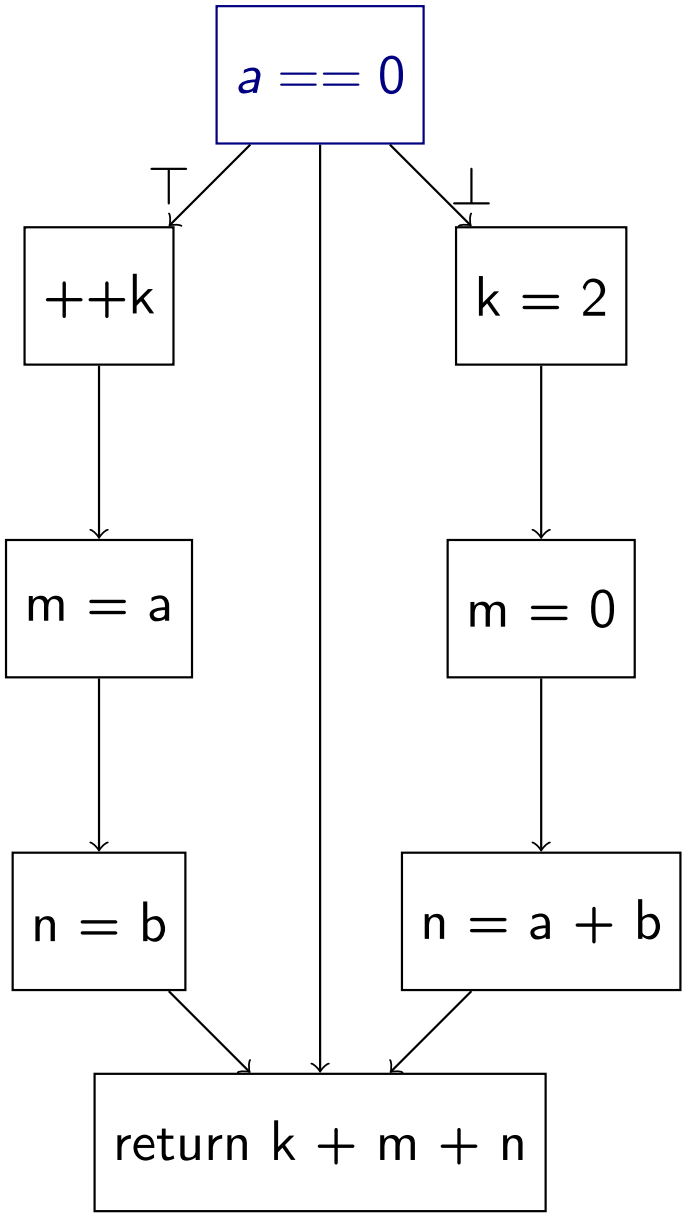
Execução concreta



Parâmetros	
a	0
b	7

Variáveis	
k	1
m	
n	
return	

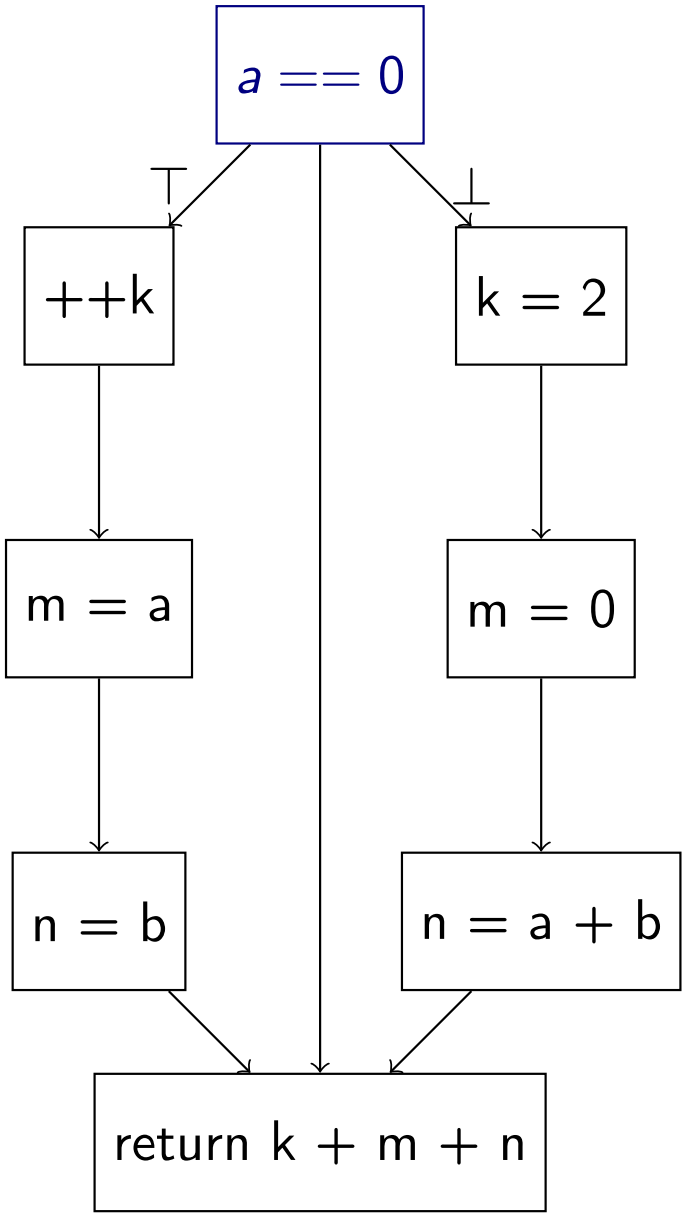
Execução concreta



Parâmetros	
a	0
b	7

Variáveis	
k	2
m	
n	
return	

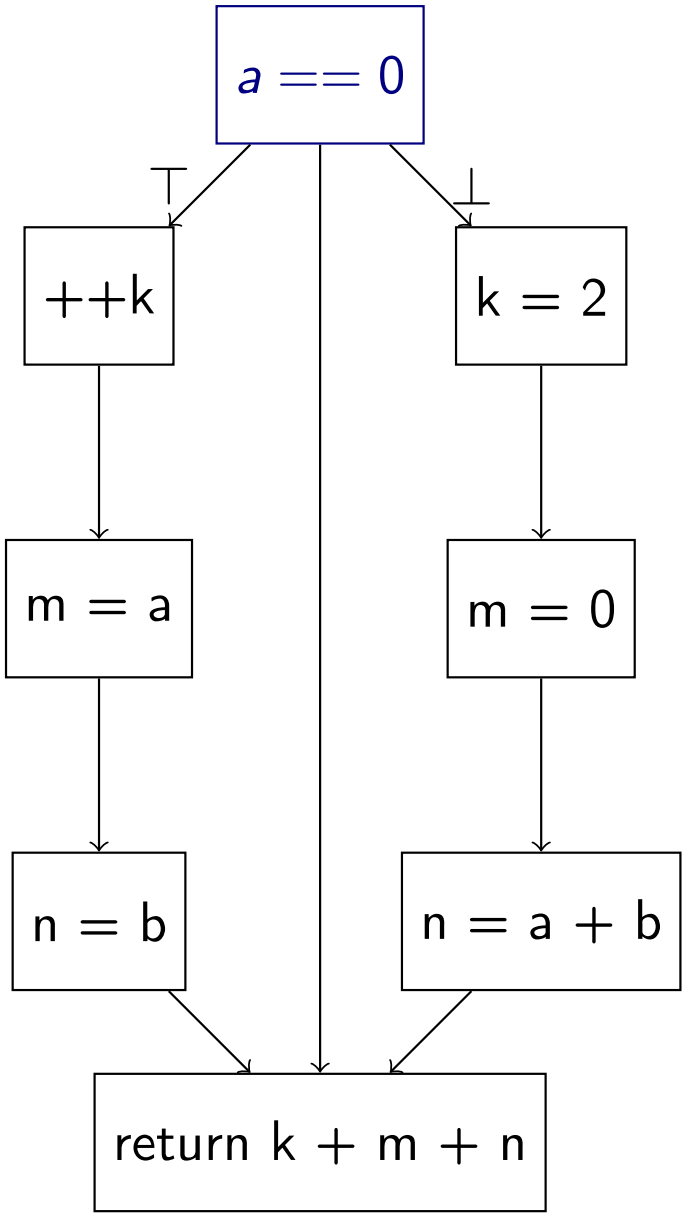
Execução concreta



Parâmetros	
a	0
b	7

Variáveis	
k	2
m	0
n	
return	

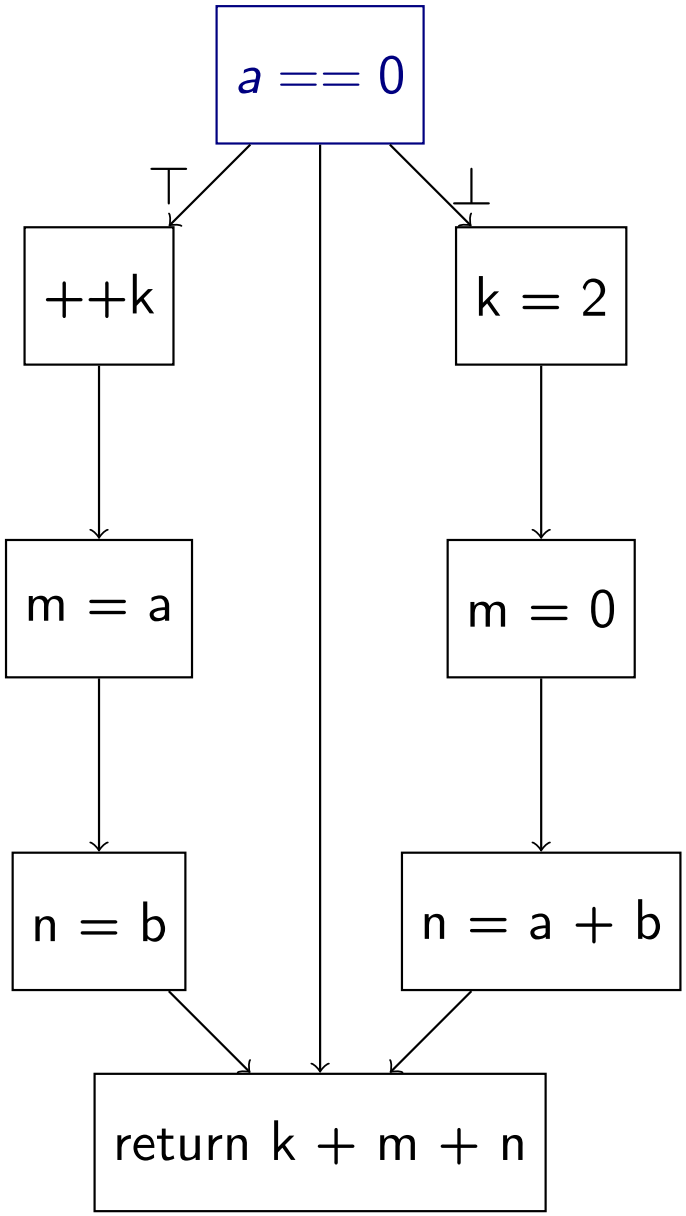
Execução concreta



Parâmetros	
a	0
b	7

Variáveis	
k	2
m	0
n	7
return	

Execução concreta



Parâmetros	
a	0
b	7

Variáveis	
k	2
m	0
n	7
return	9

Obsevações

Questão

Pode ser deduzido que $k = 2$ a partir das execuções concretas ?

Obsevações

Questão

Pode ser deduzido que $k = 2$ a partir das execuções concretas ?

- Sim mas ...
- Precisa fazer $2^{64} * 2^{64}$ testes

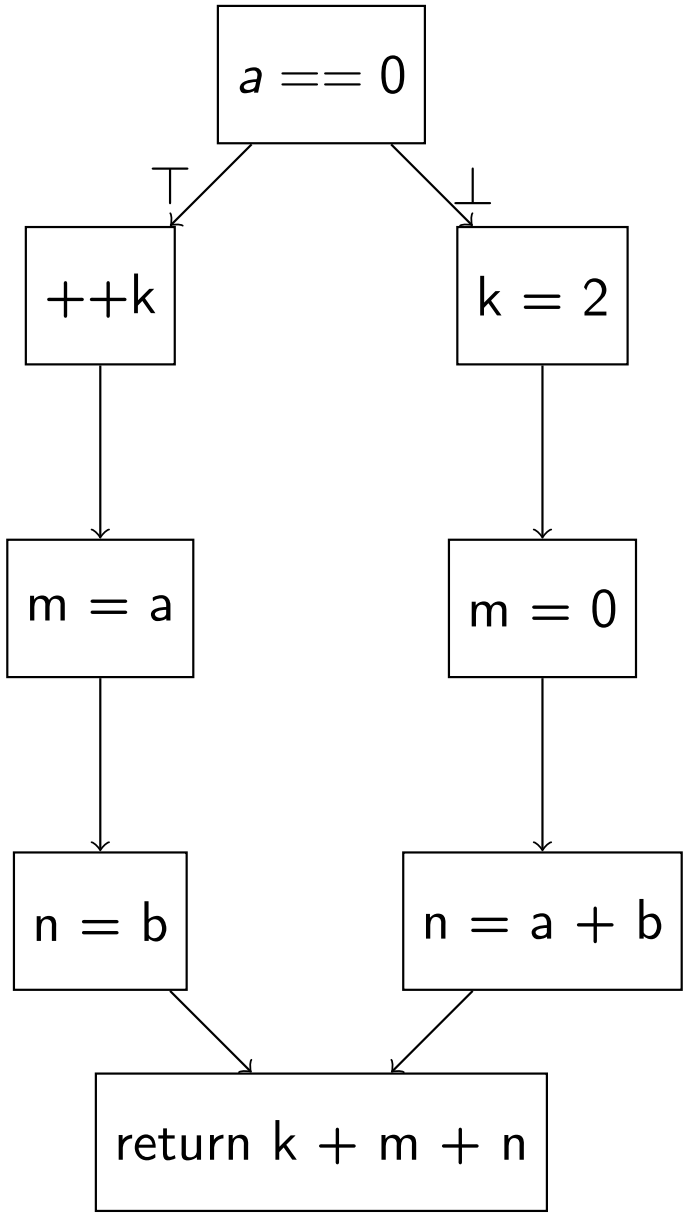
Obsevações

Questão

Pode ser deduzido que $k = 2$ a partir das execuções concretas ?

- Sim mas ...
- Precisa fazer $2^{64} * 2^{64}$ testes
- Tempo aproximativo: 10^{20} anos si um teste precisa de 10^{-11} segundos.

Execução abstrata



Sintaxe da linguagem `While`

Categorias sintáticas

- n, n_i, n' = elementos numéricos (Num)
- x = variáveis (Var)
- a = expressões aritméticas ($_e xp$)
- b = expressões booleanas ($_e xp$)
- S = instruções

BNF

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2 \\
b & ::= & \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & x := a \mid \text{skip} \mid S_1; S_2 \\
 & \mid & \text{if } b \text{ then } S_1 \text{ else} S_2 \\
 & \mid & \text{while } b \text{ do } S
\end{array}
$$

Uma linguagem aritmética

Sintaxe

$$
\begin{array}{lcll}
exp & ::= & n & \text{número} \\
 & | & exp + exp & \text{adição} \\
 & | & exp * exp & \text{multiplicação} \\
 & | & exp - exp & \text{subtração}
\end{array}
$$

Imprecisão

Às vezes é difícil determinar com precisão absoulta o **sinal** duma operação

- Por exemplo a adição de um número positivo e de um número negativo pode ser negativa ou positiva

Determinar o sinal de uma expressão

Definição (Valores possíveis)

$Sign = \{\text{zero, pos, neg, num}\}$

Adição ($\oplus$) e multiplicação ($\otimes$) abstratas

$\oplus : Sign \times Sign \rightarrow Sign$

$\oplus$	zero	pos	neg	num
zero	zero	pos	neg	num
pos	pos	pos	num	num
neg	neg	neg	num	num
num	num	num	num	num

$\otimes : Sign \times Sign \rightarrow Sign$

$\otimes$	zero	pos	neg	num
zero	zero	zero	zero	zero
pos	zero	pos	neg	num
neg	zero	neg	pos	num
num	zero	num	num	num

- Determinar a interpretação da subtração abstrata $\ominus$

Interpretação

Interpretação abstrata

Definimos uma função de interpretação

$$[\![\cdot]\!] : \mathbb{N} \to \mathit{Sign}$$

Definição

$$\begin{array}{lcl} [\![n]\!] & = & \mathit{sign}(n) \\ [\![e_1 + e_2]\!] & = & [\![e_1]\!] \oplus [\![e_2]\!] \\ [\![e_1 * e_2]\!] & = & [\![e_1]\!] \otimes [\![e_2]\!] \\ [\![e_1 - e_2]\!] & = & [\![e_1]\!] \ominus [\![e_2]\!] \end{array}$$

- $\mathit{sign}(n) \equiv$ se $n > 0$ então `pos` senão `neg`.

Exercício

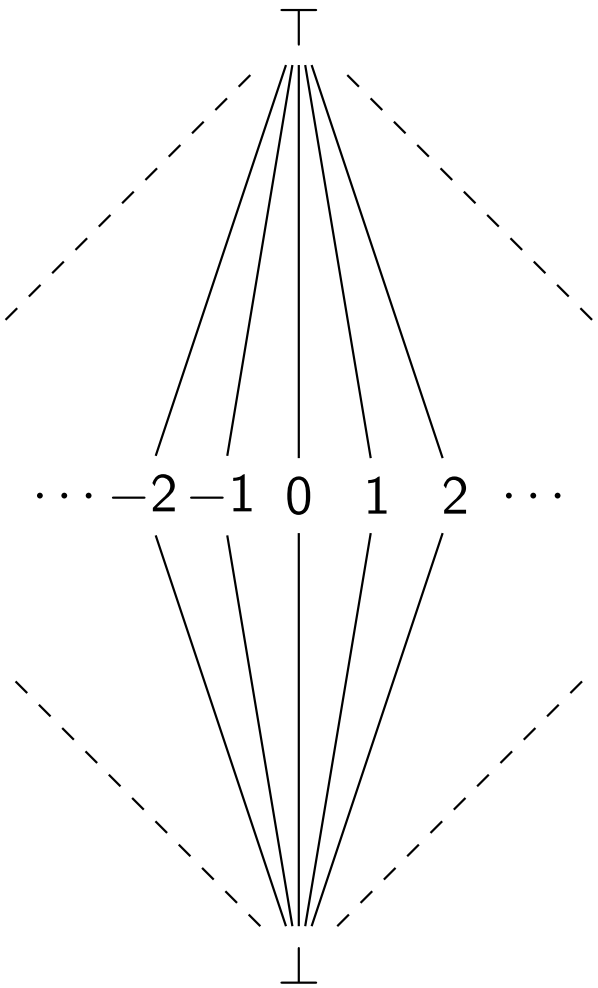
Assunto

Interpretar o programa `While` abaixo

```
x := 1 - 2 * 4
if x then y := 3 else y := -4
z := x * y
```

Propagação de constantes

Usaremos o reticulado abaixo



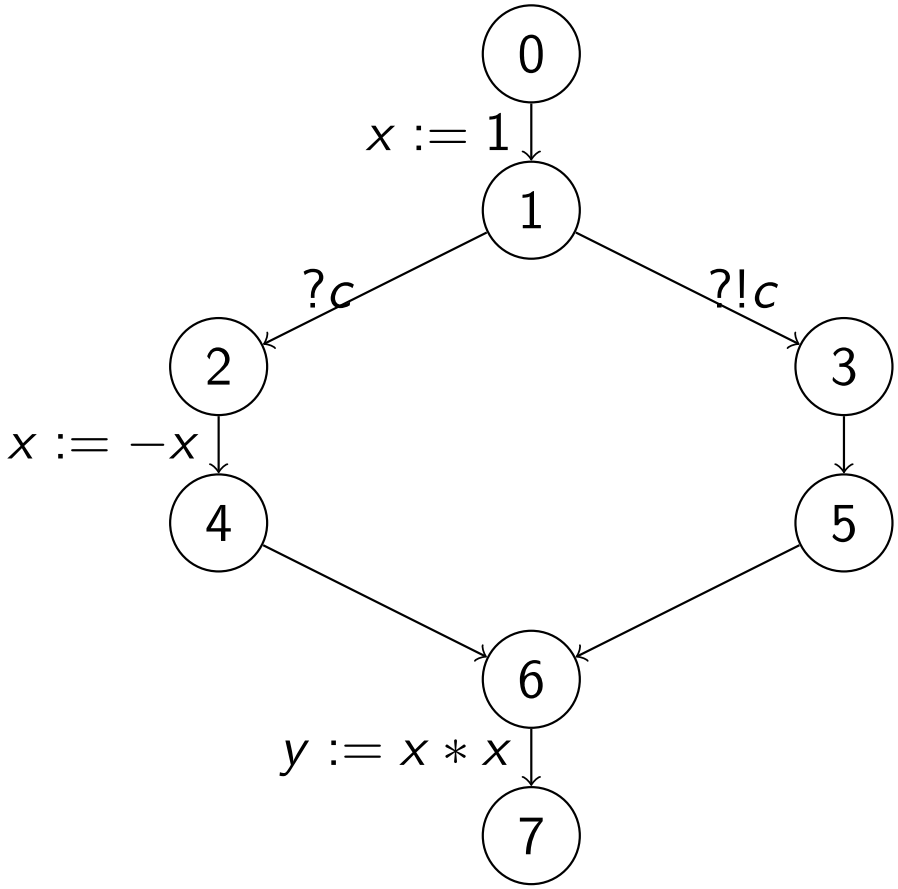
Contexto de análise

- $E = Var \rightharpoonup \mathcal{C}$
- $i = \lambda x.\top$
- $f_{x:=e}(\sigma^\sharp) = \sigma^\sharp[x \leftarrow [\![e]\!]\sigma^\sharp]$
- $f_{skip}(\sigma^\sharp) = \sigma^\sharp$
- $f_{?e}(\sigma^\sharp) = \begin{cases} \bot & \text{se } [\![e]\!]\sigma^\sharp = 0 \\ \sigma^\sharp & \text{senão} \end{cases}$

Propriedades

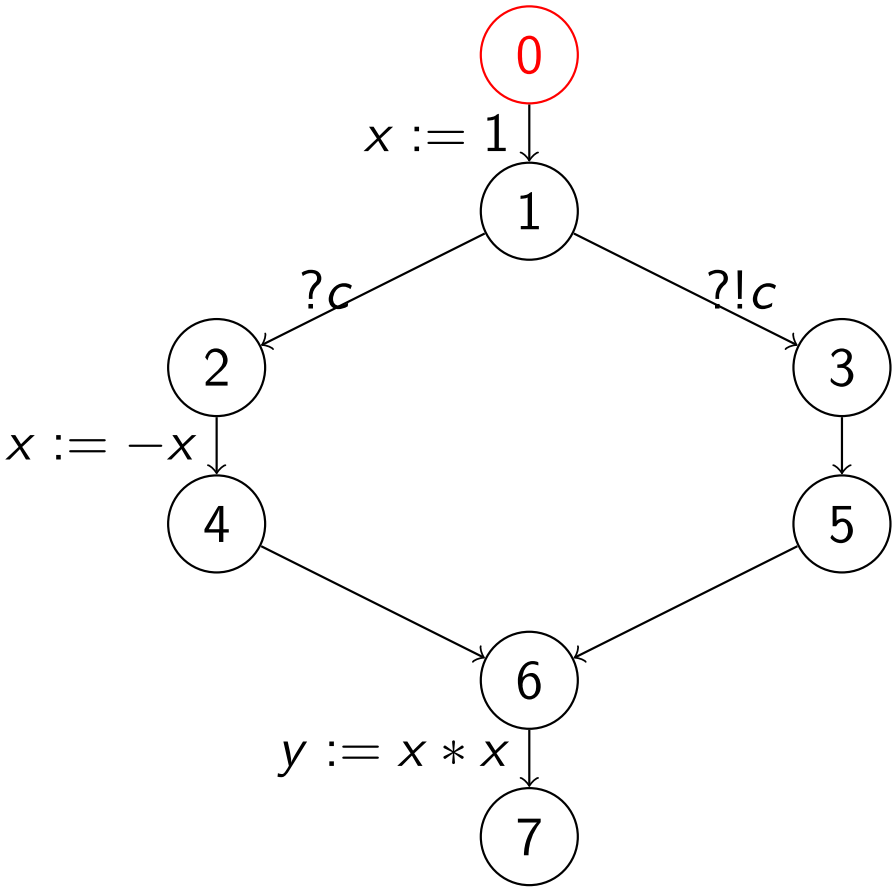
- **Corretude**: Se, no fim da análise, ${\sigma^\sharp}_i(x) = v$, todo caminho execução passando por i associa v a x.
- **Terminação**: A análise sempre termina.

Exemplo de propagação



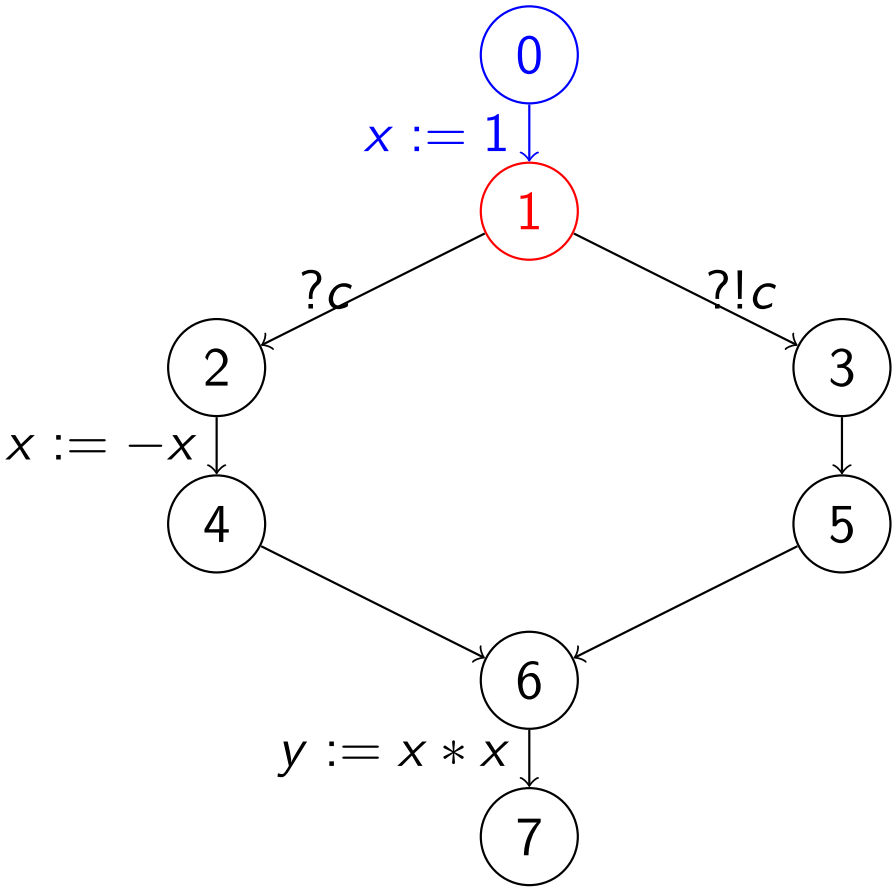
nœud	c	x	y
0			
1			
2			
3			
4			
5			
6			
7			

Exemplo de propagação



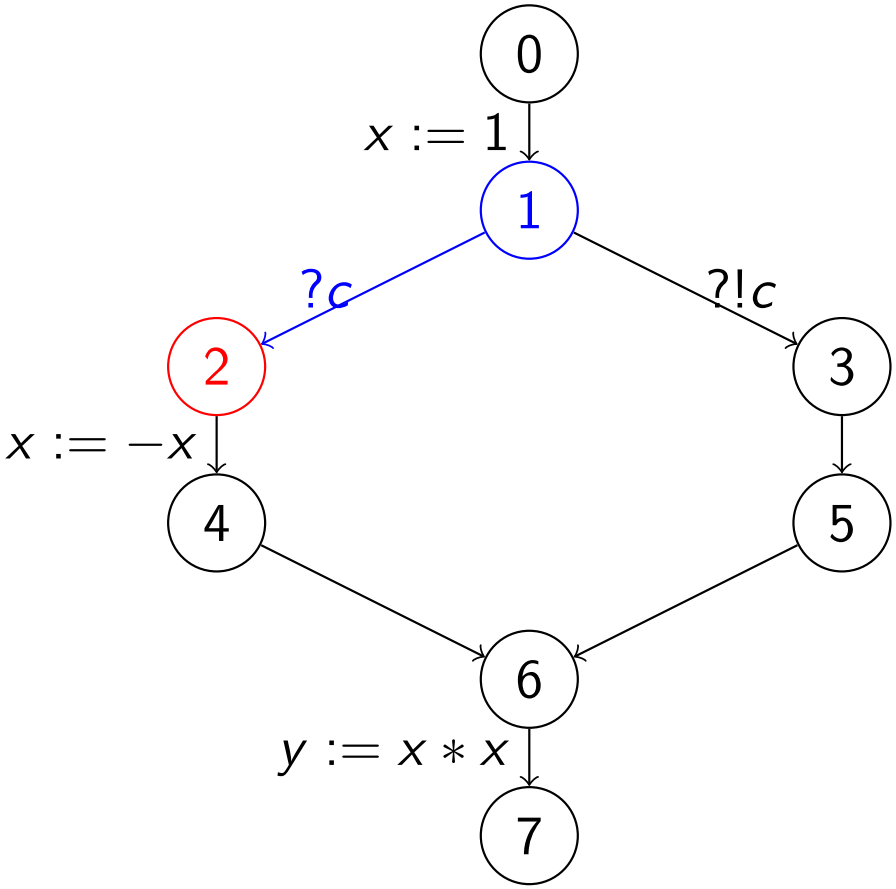
nœud	c	x	y
0	$\top$	$\top$	$\top$
1			
2			
3			
4			
5			
6			
7			

Exemplo de propagação



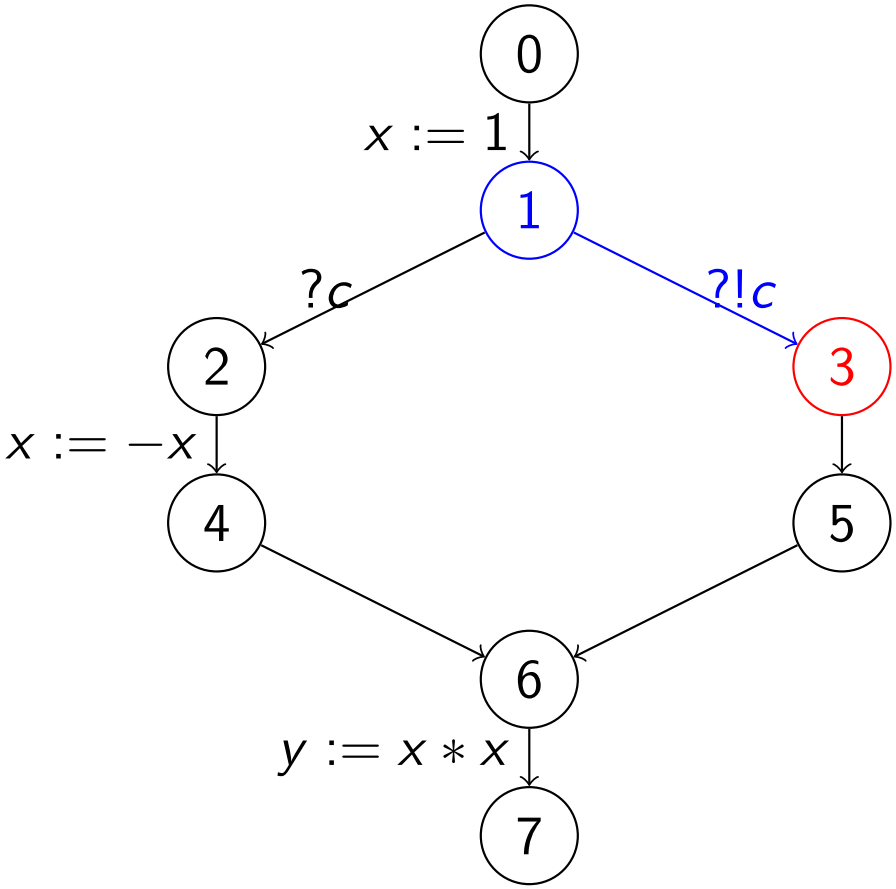
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2			
3			
4			
5			
6			
7			

Exemplo de propagação



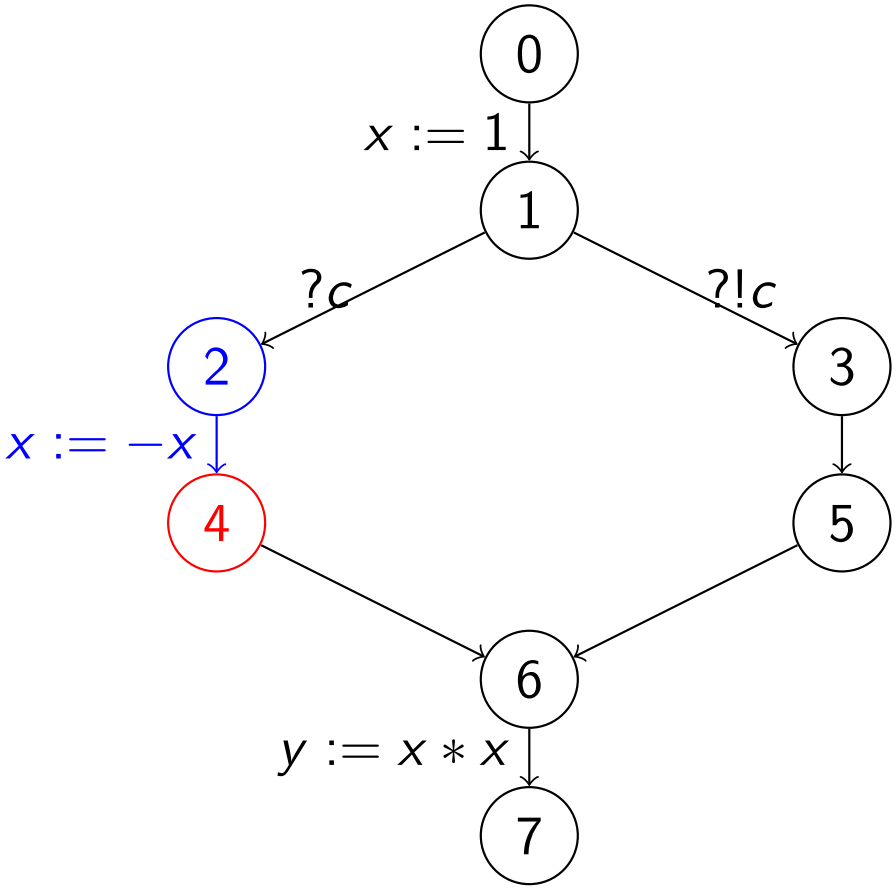
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2	$\top$	1	$\top$
3			
4			
5			
6			
7			

Exemplo de propagação



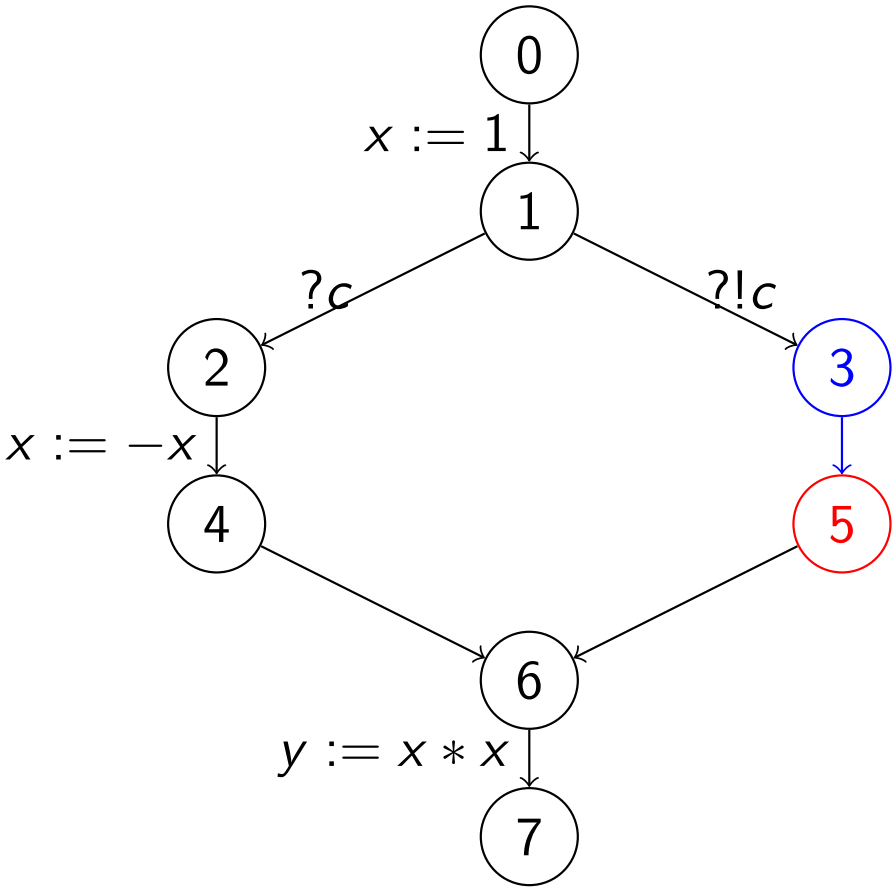
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2	$\top$	1	$\top$
3	0	1	$\top$
4			
5			
6			
7			

Exemplo de propagação



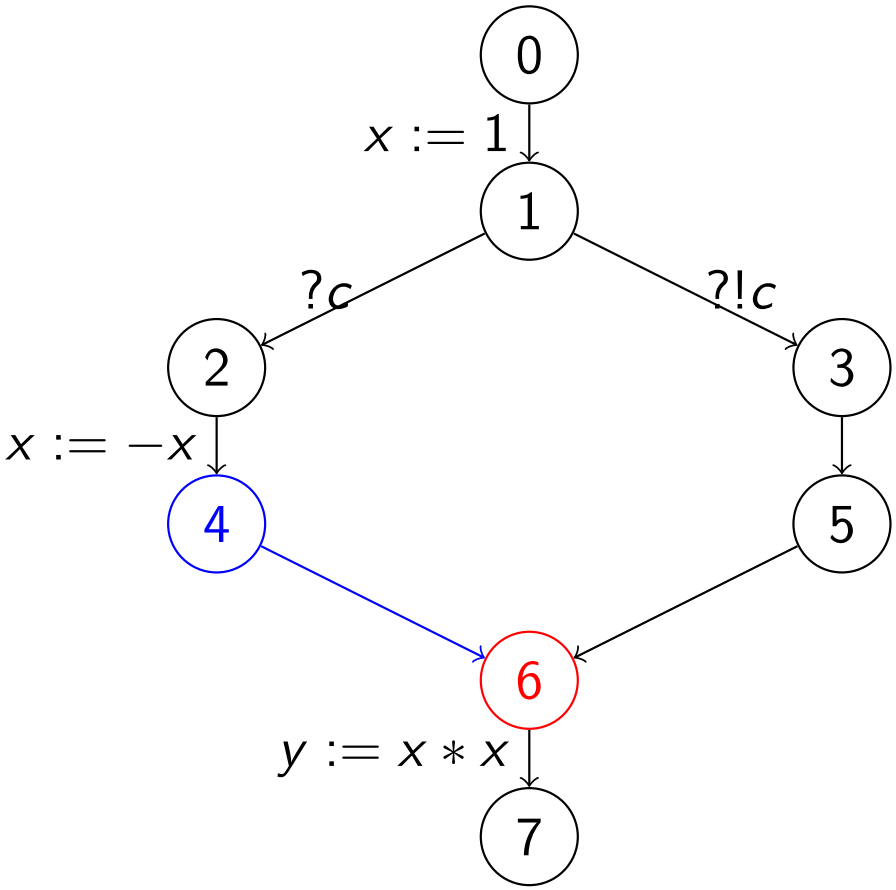
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2	$\top$	1	$\top$
3	0	1	$\top$
4	$\top$	-1	$\top$
5			
6			
7			

Exemplo de propagação



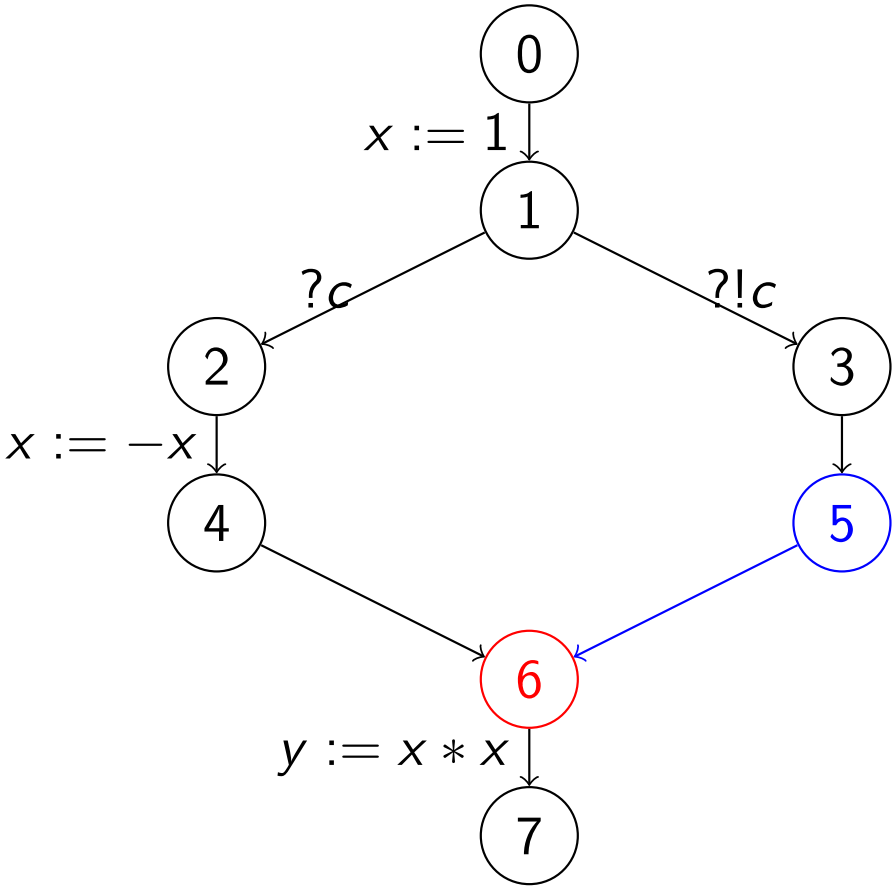
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2	$\top$	1	$\top$
3	0	1	$\top$
4	$\top$	-1	$\top$
5	0	1	$\top$
6			
7			

Exemplo de propagação



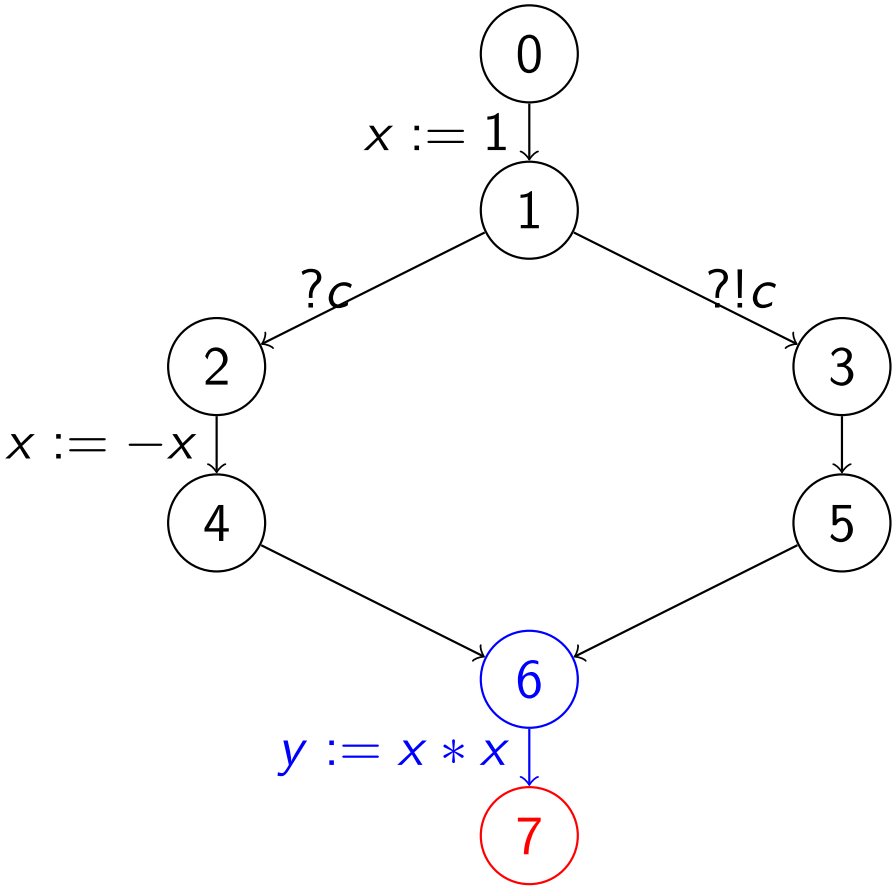
nœud	c	x	y
0	$\top$	$\top$	$\top$
1	$\top$	1	$\top$
2	$\top$	1	$\top$
3	0	1	$\top$
4	$\top$	-1	$\top$
5	0	1	$\top$
6	$\top$	-1	$\top$
7			

Exemplo de propagação



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

Exemplo de propagação



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

Exercício

Assunto

Aplicar a propagação de constantes ao programa abaixo.

```
int foo(int a, int b) {
  int k = 1;
  int m, n;
  if (a == 0) {
    ++k;
    m = a;
    n = b;
  } else {
    k = 2;
    m = 0;
    n = a + b;
  }
  return k + m + n;
}
```

Reticulado

Definição

Um reticulado A é um *poset* tal que todo par $(a, b) \in A$ tem um supremo é um ínfimo.

Vocabulário particular

- A operação *join* de a e b ($a \wedge b = sup(\{a, b\})$ define o supremo de (a, b)
- A operação *meet* de a e b ($a \vee b = inf(\{a, b\})$ define o ínfimo de (a, b)

Exemplo

- Seja $A \neq \emptyset$, $(\mathcal{P}(A), \subseteq)$ é um reticulado
  - o supremo é a união dos conjuntos
  - o ínfimo é a interseção
- Qualquer conjunto totalmente ordenado define um reticulado

Axiomas dos reticulados

Seja $a, b, c \in (A, \vee, \wedge)$

- $a \vee b = b \vee a$
- $a \wedge b = b \wedge a$
- $a \vee (b \vee c) = (a \vee b) \vee c$
- $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
- $a \vee (a \wedge b) = a$
- $a \wedge (a \vee b) = a$
- $a \vee a = a$
- $a \wedge a = a$

Reticulado completo

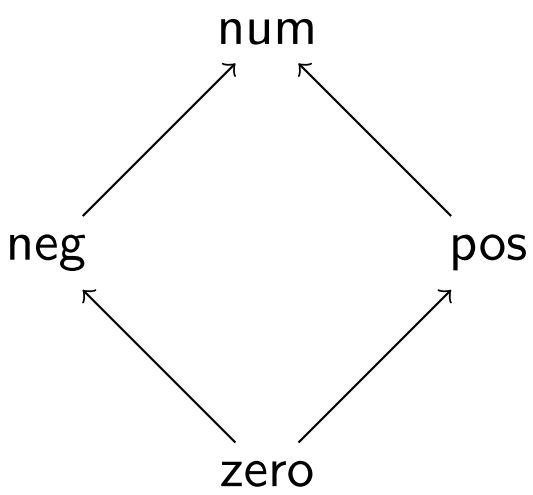
Definição

Um reticulado $(A, \vee, \wedge)$ é **completo** se $\forall B \subseteq A, \bigvee B$ e $\bigwedge B$ existem.

Teorema (Knaster-Tarski)

- *Seja $(A, \vee, \wedge)$ um reticulado completo é $f : A \to A$ uma função crescente.*
- *O conjunto de pontos fixos de f em A não é vazio e é um reticulado completo.*
- *f tem um menor é um maior ponto fixo em A*

Reticulado dos sinais



Abstração e concretização

Abstração

Uma função de **abstração** α é um mapeamento de um objeto concreto o para uma aproximação do domínio de interpretação $\alpha(0)$.

Concretização

Uma função de **concretização** γ é um mapeamento de um objeto abstrato $\overline{o}$ para uma um objeto concreto $\gamma(\overline{o})$.

Abstração do reticulado dos sinais

Definição

- $\gamma : \mathit{Sign} \to \mathcal{P}(\mathbb{Z}) \backslash \{\emptyset\}$

$$
\begin{array}{lcl}
\gamma(\mathsf{zero}) & = & \{0\} \\
\gamma(\mathsf{pos}) & = & \{x \mid x > 0\} \\
\gamma(\mathsf{neg}) & = & \{x \mid x < 0\} \\
\gamma(\mathsf{num}) & = & \mathbb{Z}
\end{array}
$$

Concretização do reticulado dos sinais

Definição

- $\alpha : \mathcal{P}(\mathbb{Z})\backslash\{\emptyset\} \rightarrow \mathit{Sign}$

$$
\begin{array}{lcll}
\alpha(\{0\}) & = & \text{zero} & \\
\alpha(X) & = & \text{pos} & \text{si } \forall x \in X > 0 \\
\alpha(X) & = & \text{neg} & \text{si } \forall x \in X < 0 \\
\alpha(X) & = & \text{num} & \text{dans les autres cas}
\end{array}
$$

Exemplo

$$
\begin{array}{rcl}
\alpha(\{2,3,1\}) & = & \text{pos} \\
\alpha(\{-1,-2,-3\}) & = & \text{neg} \\
\alpha(\{1,2,-4\}) & = & \text{num}
\end{array}
$$

Teorema

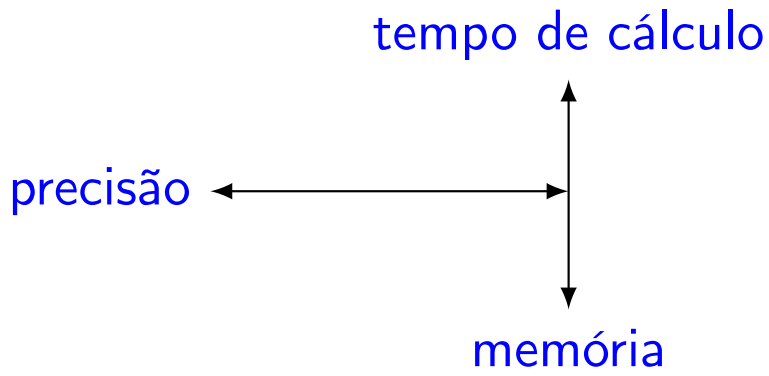
Teorema (Segurança da abstração)

$\forall e \; \{E_{std}[\![e]\!]\} \subseteq \gamma(E_{ros}[\![e]\!])$

Como escolher o seu domínio abstrato ?

Na prática, escolher o domínio abstrato é **fundamental**

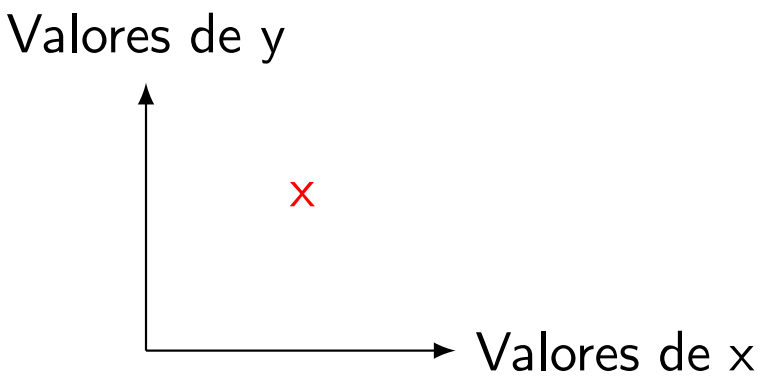
- deve ser suficientamente **preciso**
- em particular, deve permitir expressar a propriedade desejada
- deve ser calculável com o custo **tempo/memória razoavel**
  - i.e. horas de cálculo, Gb de RAM para casos reais



- **Domínio não relacional**: nenhuma relação entre elemento é conservada. Pouco preciso mas pouco custoso
- **Domínio relacional**: relações entre elenentos do domínio. Mais préciso mas custa

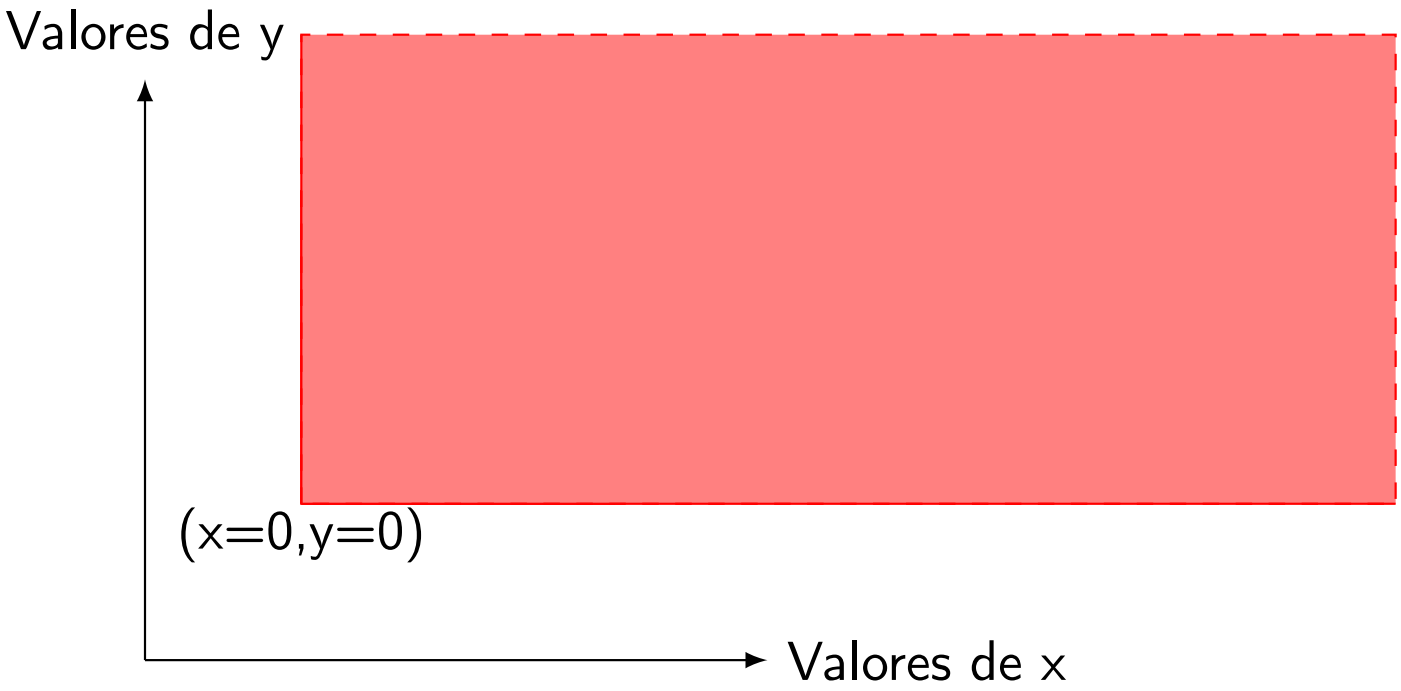
Domínio das constantes

- $x = z$ $(z \in \mathbb{Z})$
- domínio não relacional
- se o valor exato não é conhecido, perde a informação inteira



Domínio dos sinais

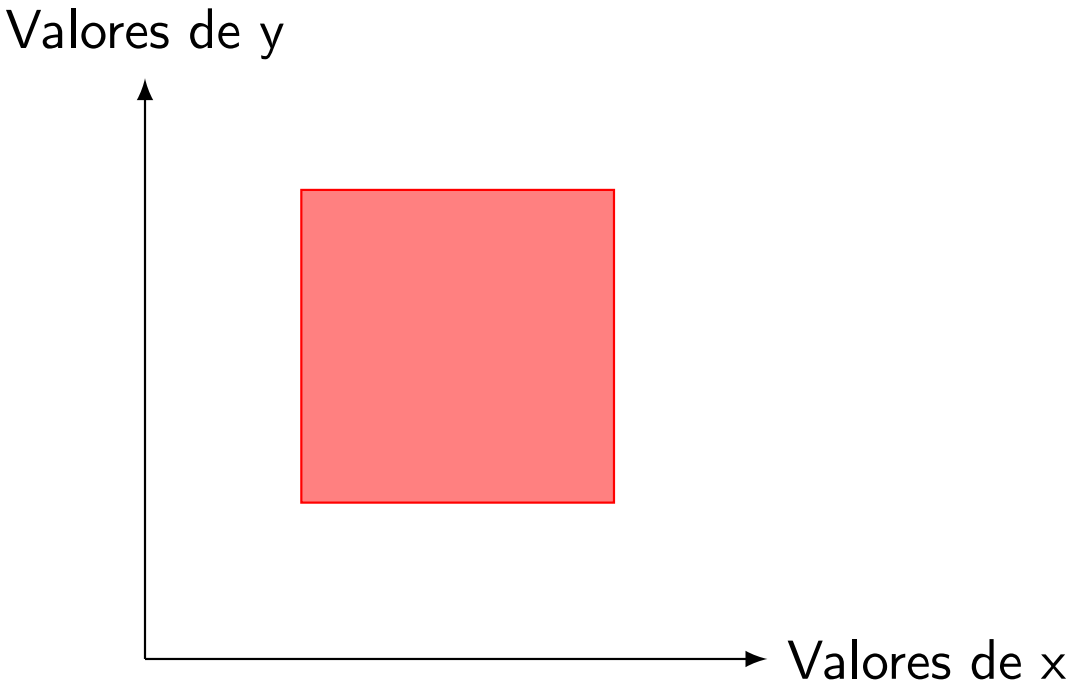
- $x \text{op} 0$, avec op $\in \{\geq, >, \leq, <, =, \neq\}$
- domínio não relacional
- conservação dos valores possíveis



Domínio dos intervalos

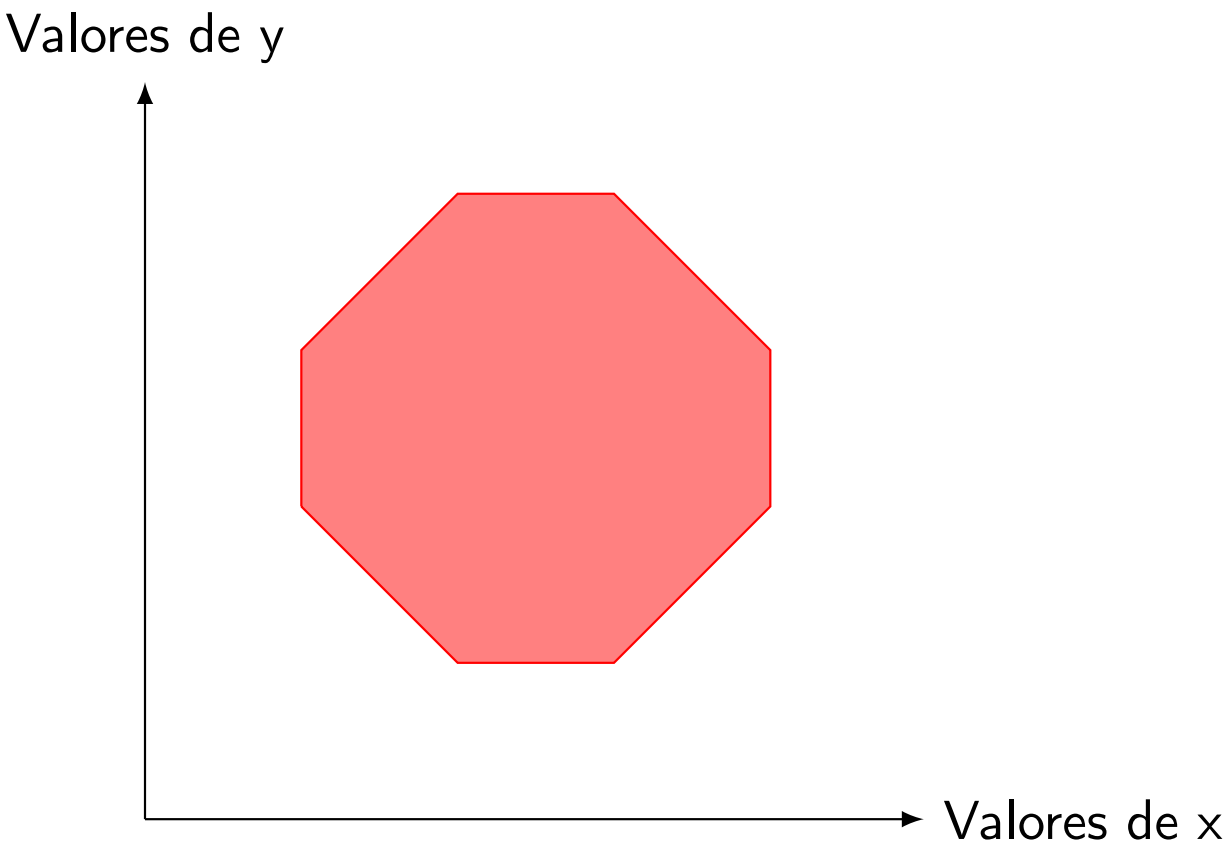
*

- $x \in [i_0, i_1]$
- domínio não relacional
- conservação de um intervalo agrupando todos os valores possíveis



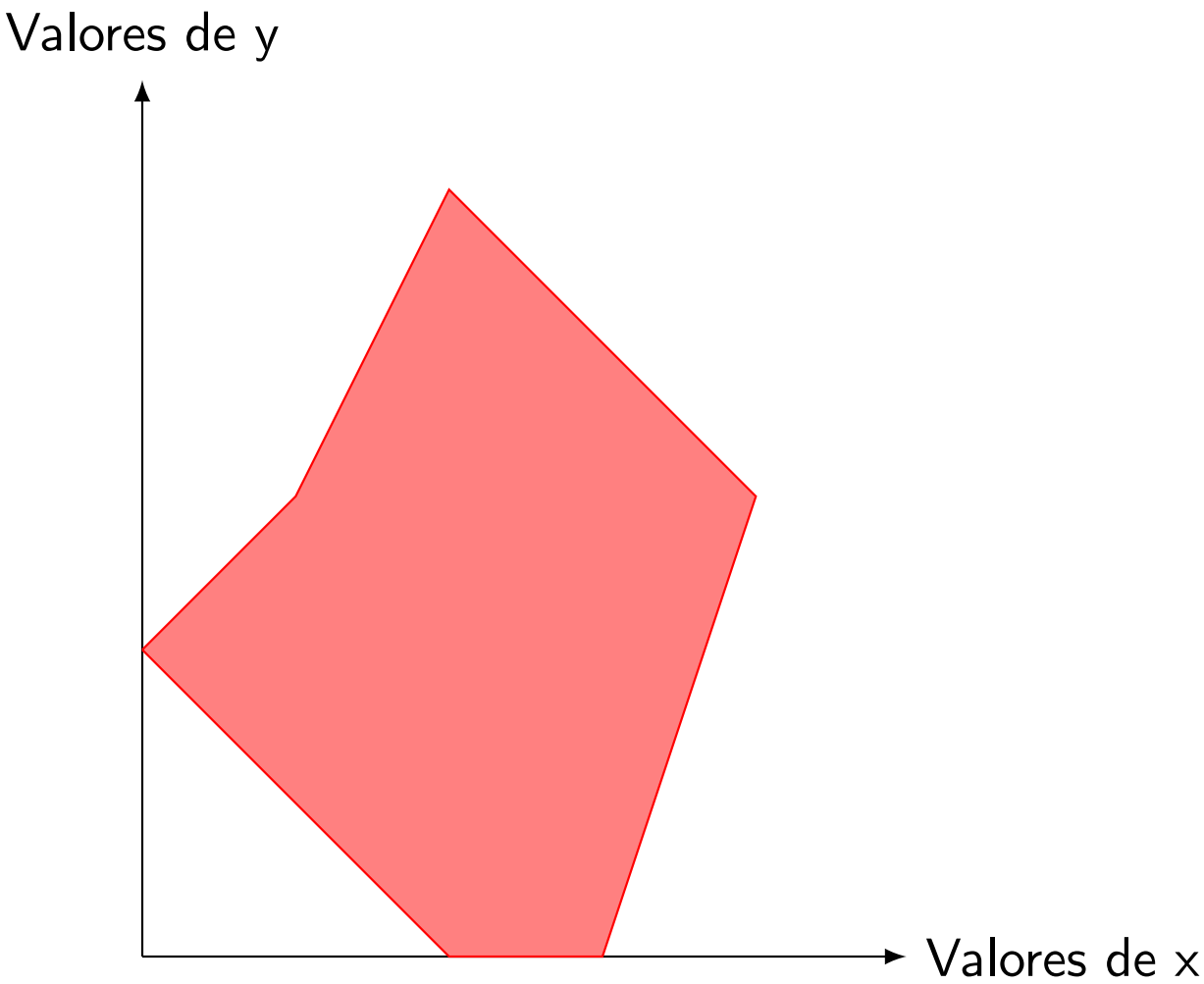
Domínio dos octogonos

- $\pm x \pm y \leq c$
- domínio relacional
- conservação de relações lineares simples entre elementos



Domínio dos políedros

- $kx + ly \leq c$
- domínio relacional
- relações lineares complexas entre elementos



Análise de valor

Descrição

- Análise por interpretação abstrata de programas **sequenciais**
- Cálculo dos domínios de variação das variáveis do programa
- Inferência da ausência de error de execução

Resumo

Referências

Patrick Cousot and Radhia Cousot, *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*, Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (New York, NY, USA), POPL '77, ACM, 1977, pp. 238–252.

Flemming Nielson, Hanne Riis Nielson, and Chris Hankin, *Principles of program analysis (2. corr. print)*, Springer, 2005.

Perguntas ?

http://dimap.ufrn.br/~richard/dim0436