

# Análisis Automático de Programas

## Interpretación abstracta

Diego Garbervetsky  
Departamento de Computación  
FCEyN – UBA

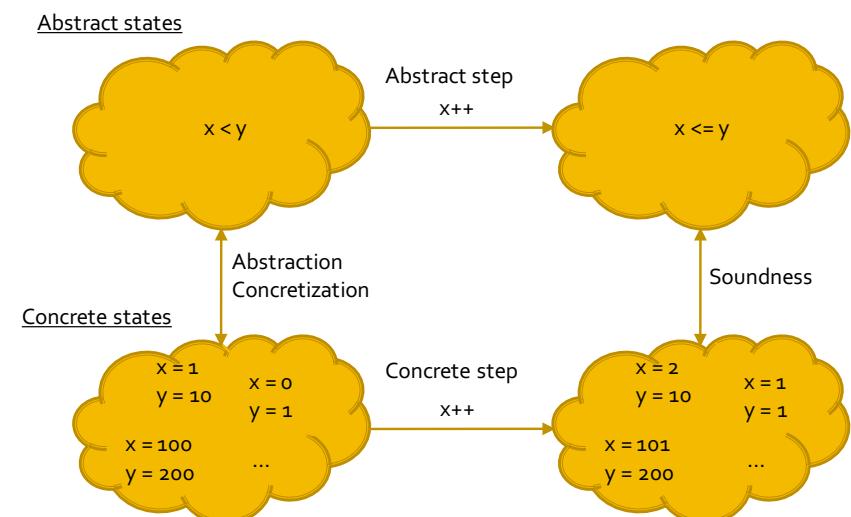
### Interpretación abstracta by Cousot

- **Static analysis:** analyze the program at compile-time to verify a program runtime property (e.g. the absence of some categories of bugs)
  - Undecidability
- **Abstract interpretation:** effectively compute an abstraction/**sound approximation** of the program semantics,
  - which is **precise** enough to imply the desired property
  - coarse enough to be **efficiently computable**.

# Abstract Interpretation



### Abstract Interpretation (on 1 slide)



## Interpretación abstracta

- Una metodología general para el diseño de analizadores estáticos de programas.
  - Correcta por construcción
  - Genérica
  - Fácil de ajustar

## Interpretación Abstracta

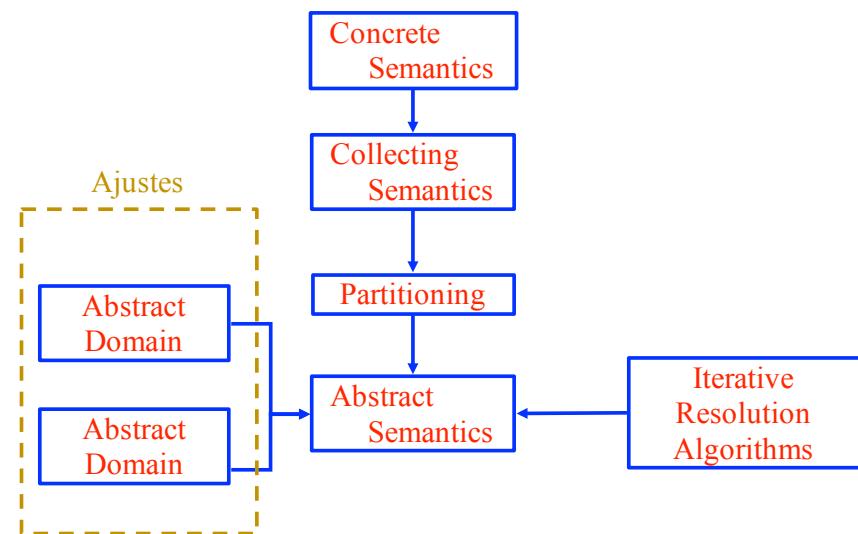
- Comenzamos con una especificación formal de la semántica del programa: la **semántica concreta**
- Construimos ecuaciones que representen la semántica abstracta con respecto a algún esquema de aproximación
- Usamos algoritmos generales para resolver las ecuaciones de semántica (puntos fijos)
- Probamos diferentes instanciaciones de los esquemas para ver cual es el que mejor ajusta

## Aproximación

La idea principal del framework de Interpretación Abstracta es la de formalizar la noción de **aproximación**

- Se da una noción de una aproximación del estado
- Se define una aproximación para las operaciones atómicas
- La aproximación se extiende automáticamente a toda la estructura del programa

## Metodología



## Reticulados y puntos fijos

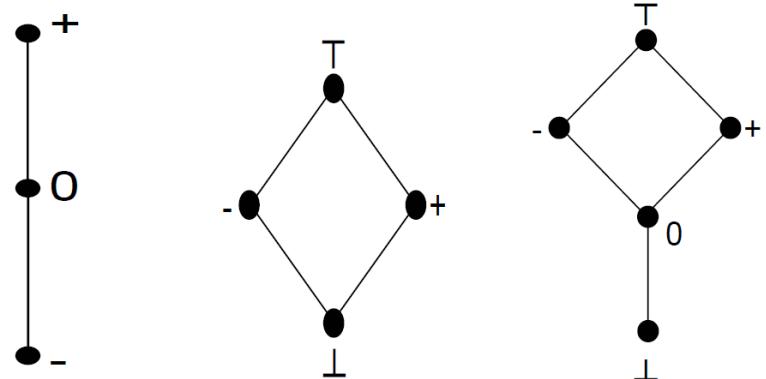
- Un **reticulado**  $(L, \sqsubseteq, \perp, \sqcup, \top, \sqcap)$  es un conjunto parcialmente ordenado  $(L, \sqsubseteq)$  con:
  - Operadores de supremo ( $\sqcup$ ) e infímo ( $\sqcap$ )
  - Un elemento mínimo “bottom”:  $\perp$
  - Un elemento máximo “top”:  $\top$
- $L$  es completo si todos los supremos existen

## Puntos fijos

- Un punto fijo  $x$  de  $F: L \rightarrow L$  satisface  $F(x) = x$
- Sea  $FP_F = \{x \in L \mid F(x) = x\}$
- Notamos como **lfp F** al menor punto fijo, si este existe.
  - $lfp F = \sqcap FP_F$
- Notamos como **gfp F** al mayor punto fijo, si este existe.
  - $gfp F = \sqcup FP_F$

## Reticulados

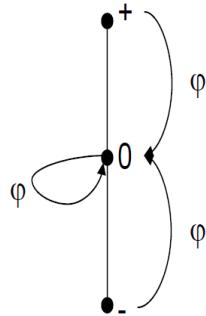
$(\mathbb{Z} \cup \{-\infty, \infty\}, \leq, -\infty, \infty, \max, \min)$   
 $(\wp(\mathbb{Z}), \subseteq, \emptyset, \mathbb{Z}, \cup, \cap)$



## Propiedades

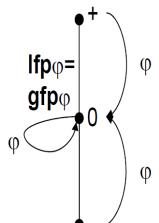
- $F: L \rightarrow L$  es **monotona** si
  - $\forall x, y \in L, x \sqsubseteq y \rightarrow F(x) \sqsubseteq F(y)$
- $F$  es **upper continua** (preserva supremo) si
  - $\forall X = \{x_1 \sqsubseteq \dots \sqsubseteq x_n\} F(\sqcup X) = \sqcup F(X)$ 
    - $\sqcup F(X)$  representa  $F(x_1) \sqcup \dots \sqcup F(x_n)$
- $F$  es **continua** si cumple son estas dos propiedades

## Ejemplo

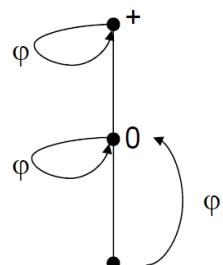


$$\begin{aligned}\varphi(\sqcup\{-\}) &= 0 = \sqcup\varphi- \\ \varphi(\sqcup\{-, 0\}) &= 0 = \sqcup\varphi\{-, 0\} \\ \varphi(\sqcup S) &= 0 = \sqcup\varphi S\end{aligned}$$

## Ejemplo

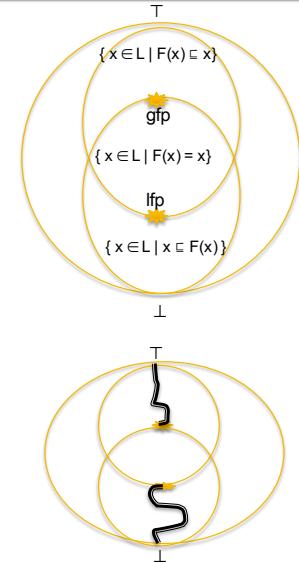


$$\begin{aligned}X_0 &= \perp = - \\ X_1 &= \varphi X_0 = \varphi- = 0 \\ X_2 &= \varphi X_1 = \varphi 0 = 0 = \text{lfp}\varphi\end{aligned}$$



## Teoremas del punto fijo

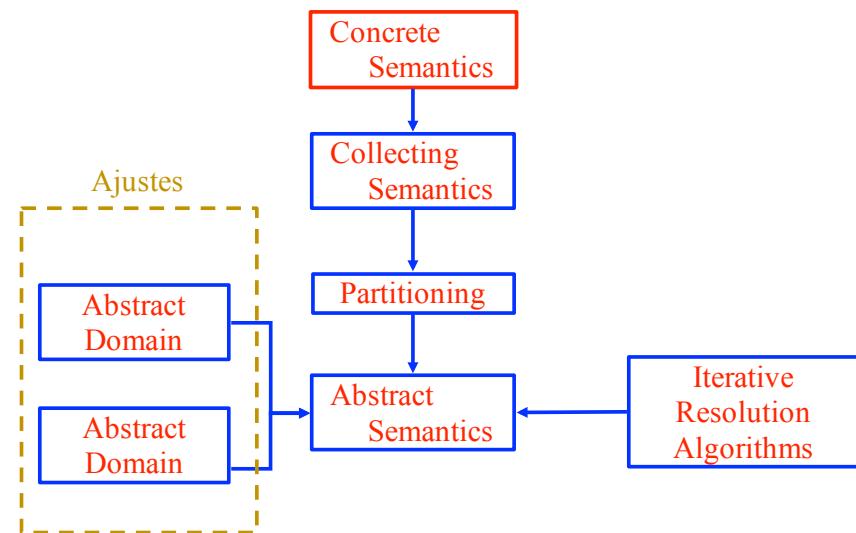
- Teorema de Knaster-Tarski: Si  $F: L \rightarrow L$  es monótona y  $L$  es un reticulado completo, el conjunto  $\text{FP}_F$  de puntos fijos de  $F$  es también un reticulado completo y  $\text{lfp} F = \sqcap \text{FP}_F$



- Teorema de Kleene: Si  $F: L \rightarrow L$  es continua,  $L$  es un reticulado completo entonces  $\text{lfp } F$  se define como:

$$\sqcup \left\{ \begin{array}{l} F_0 = \perp \\ F_{n+1} = F(F_n) \end{array} \right.$$

## Metodología



## Semantica Concreta

Semántica operacional small-step:  $(S, \rightarrow)$

$$s = \langle \text{program point}, \text{env} \rangle$$

$$s \rightarrow s'$$

Ejemplo:

```
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit
```

$$\langle 1, n \Rightarrow \perp \rangle \rightarrow \langle 2, n \Rightarrow 0 \rangle \rightarrow \langle 3, n \Rightarrow 0 \rangle \rightarrow \langle 4, n \Rightarrow 1 \rangle \\ \rightarrow \langle 2, n \Rightarrow 1 \rangle \rightarrow \dots \rightarrow \langle 5, n \Rightarrow 1000 \rangle$$

Valor indefinido concreto

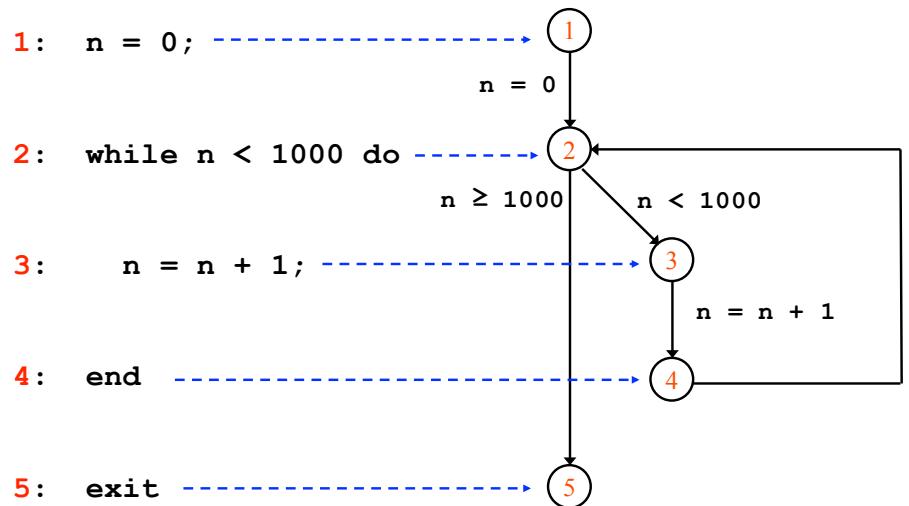
## Relación de transición

Control flow graph:  $i \xrightarrow{\text{op}} j$

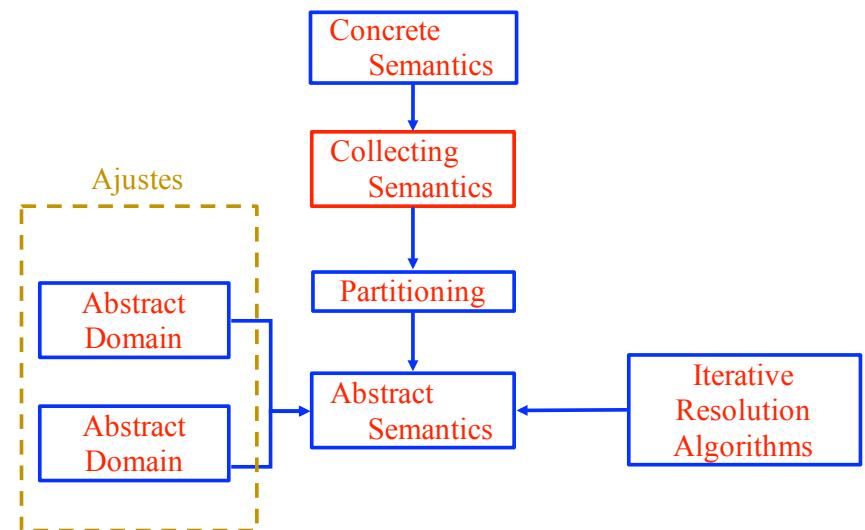
Semántica operacional:  $\langle i, \varepsilon \rangle \rightarrow \langle j, [\text{op}] \varepsilon \rangle$

Semántica de op

## Control Flow Graph



## Metodología



# Collecting Semantics

La semántica de “recolección” (collecting semantics) representa un **conjunto de comportamiento observable** en la semántica operacional.

## Ejemplos

- El conjunto de estados alcanzables desde el estado inicial
- El conjunto de todos los estados alcanzables desde el estado inicial que alcanzan un estado final
- El conjunto de todas las trazas finitas desde el estado inicial
- El conjunto de todas las trazas finitas e infinitas desde el estado inicial

**Intuición:** Colecta los valores que fluyen por el CFG

## Propiedades sobre el estado

El conjunto de estados alcanzables desde el estado inicial  $s_0$ :

$$S = \{s \mid s_0 \rightarrow \dots \rightarrow s\}$$

Teorema:  $F : (\wp(S), \subseteq) \rightarrow (\wp(S), \subseteq)$

$$F(S) = \{s_0\} \cup \{s' \mid \exists s \in S : s \rightarrow s'\}$$

$$S = \text{lfp } F$$

## Qué recolectamos?

Depende del análisis

- Propiedades de estado
  - Div por cero, Buffer overuns, Nullness
- Propiedades sobre trazas finitas
  - Deadlocks, variables no inicializadas.
- Propiedades sobre trazas infinitas
  - Terminación

## Ejemplo

```
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit
```

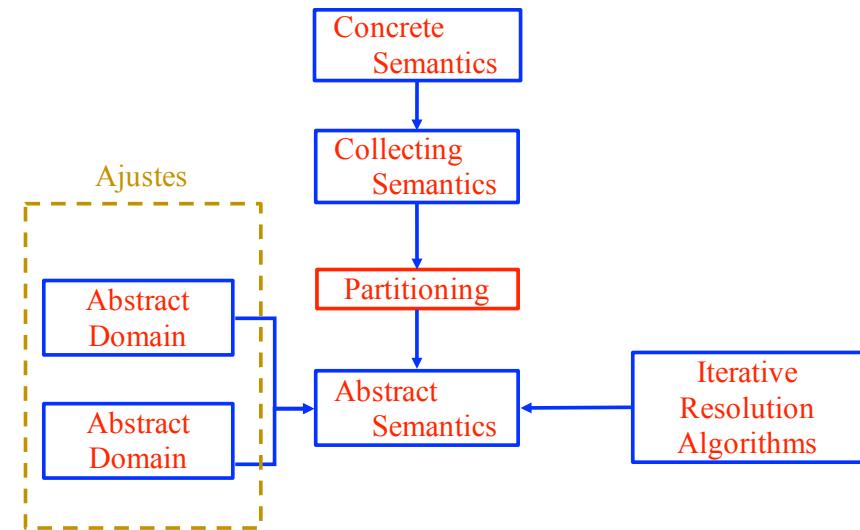
$$S = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle, \langle 4, n \Rightarrow 1 \rangle, \langle 2, n \Rightarrow 1 \rangle, \dots, \langle 5, n \Rightarrow 1000 \rangle\}$$

## Cálculo del funcional

- $F_0 = \perp = \emptyset$
- $F_1 = F(\emptyset) = \{\langle 1, n \Rightarrow \perp \rangle\}$
- $F_2 = F(F(\emptyset)) = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle\}$
- $F_3 = F(F_2) = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle\}$
- $F_4 = F(F_3) = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle, \langle 4, n \Rightarrow 1 \rangle\}$
- $F_5 = F(F_4) = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle, \langle 4, n \Rightarrow 1 \rangle, \langle 2, n \Rightarrow 1 \rangle\}$
- ...

$$S = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle, \langle 4, n \Rightarrow 1 \rangle, \langle 2, n \Rightarrow 1 \rangle, \dots, \langle 5, n \Rightarrow 1000 \rangle\}$$

## Metodología



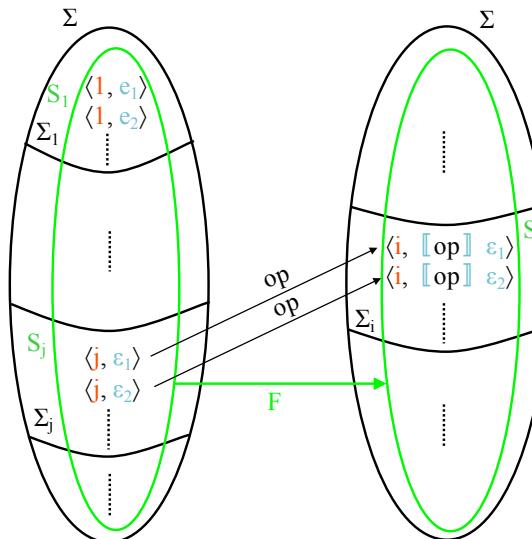
## Partitioning

Particionamos el conjunto  $S$  de estados con respecto a los puntos del programa

- Objetivo: Colectar información en cada punto del programa

- $\Sigma = \Sigma_1 \oplus \Sigma_2 \oplus \dots \oplus \Sigma_n$
- $\Sigma_i = \{\langle k, \varepsilon \rangle \in \Sigma \mid k = i\}$
- Analizamos el funcional sobre los estados particionados
  - $F_i(S_1, \dots, S_n) = \{s' \in S_i \mid \exists j \exists s \in S_j : s \xrightarrow{\text{op}} s'\}$
  - $F_i(S_1, \dots, S_n) = \{\langle i, [\text{op}] \varepsilon \rangle \mid \text{J} \xrightarrow{\text{op}} i \in \text{CFG}(P)\}$
  - $F_o(S_1, \dots, S_n) = \{s_o\}$

## Idea



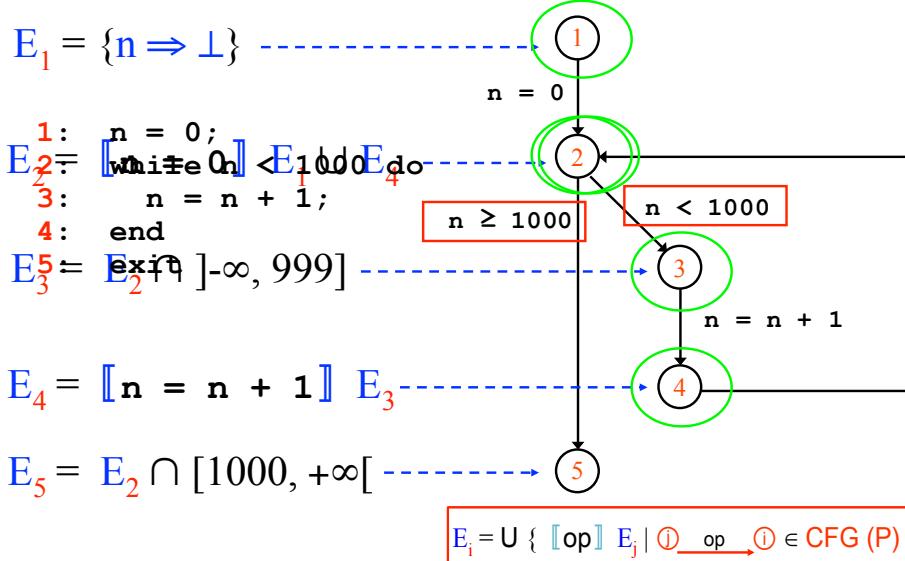
## Ecuaciones de semántica

- Notación:  $E_i = \text{conjunto de valuaciones en el punto del programa } i$ 
  - Proyectar 2<sup>da</sup> componente de  $\{\langle i, \varepsilon \rangle\}$
- Sistema de ecuaciones de semántica :

$$E_i = \bigcup \{ \llbracket \text{op} \rrbracket E_j \mid \overset{i}{\circlearrowleft} \xrightarrow{\text{op}} \overset{j}{\circlearrowright} \in \text{CFG}(P) \}$$

- Solución del sistema:  $S = \text{lfp } F$ 
  - Recordar que podemos calcular  $S$  como
    - $S^0 = (\perp, \dots, \perp)$ ,  $S^{n+1} = F(S^n) = (S_1^n, \dots, S_k^n)$
  - Usando  $E$ 
    - $E^0 = (\perp, \dots, \perp)$ ,  $E^{n+1} = \bigcup \{ \llbracket \text{op} \rrbracket E_j^n \mid \overset{n}{\circlearrowleft} \xrightarrow{\text{op}} \overset{j}{\circlearrowright} \in \text{CFG}(P) \}$

## Ejemplo



## Ejemplo

$1: n = 0;$   
 $2: \text{while } n < 1000 \text{ do}$   
 $3: \quad n = n + 1;$   
 $4: \text{end}$   
 $5: \text{exit}$

$$E_i = \bigcup \{ \llbracket \text{op} \rrbracket E_j \mid \overset{i}{\circlearrowleft} \xrightarrow{\text{op}} \overset{j}{\circlearrowright} \in \text{CFG}(P) \}$$

$$E_1 = \{n \Rightarrow \perp\}$$

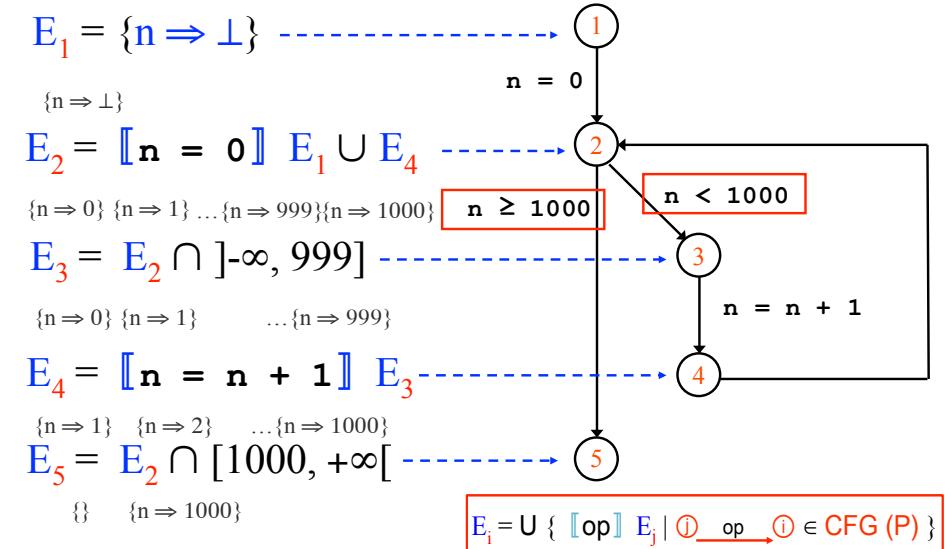
$$E_2 = \llbracket n = 0 \rrbracket E_1 \cup E_4$$

$$E_3 = E_2 \cap ]-\infty, 999]$$

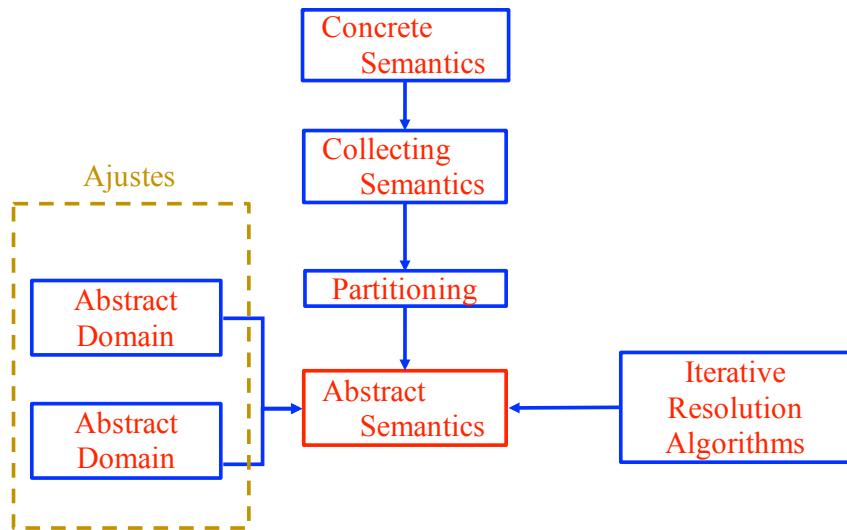
$$E_4 = \llbracket n = n + 1 \rrbracket E_3$$

$$E_5 = E_2 \cap [1000, +\infty[$$

## Ejemplo

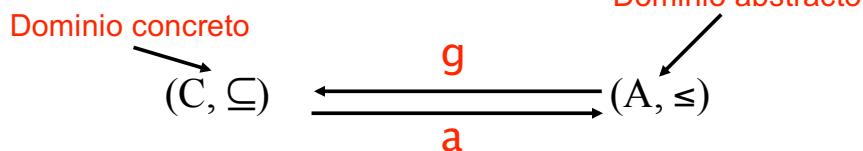


# Metodología



# Conexiones de Galois

C, A reticulados



- $\forall c \forall a : a(c) \leq a \Leftrightarrow c \subseteq g(a)$
- $\forall c \forall a : c \subseteq g \circ a(c) \text{ & } a \circ g(a) \leq a$

Si  $a \circ g(a) = a$  se lo conoce como inserción de Galois

# Aproximación

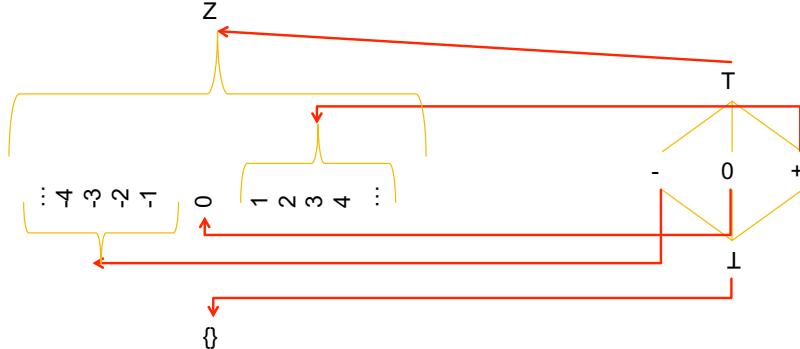
Problema: Calcular una aproximación sound de  $S: S^\#$

$$S \subseteq S^\#$$

Solución: conexiones de Galois

# Ejemplo

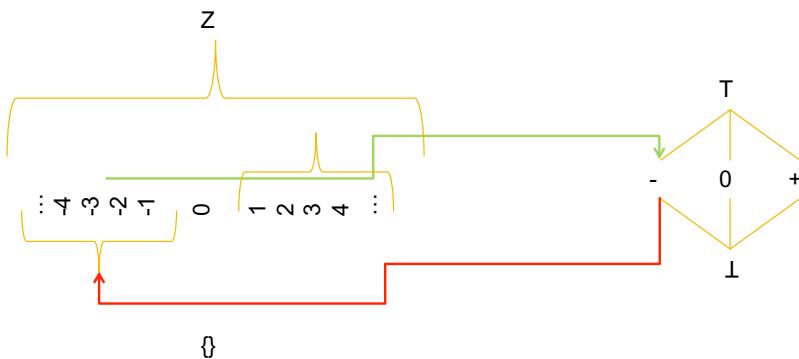
- $\forall c \forall a : a(c) \leq a \Leftrightarrow c \subseteq g(a)$
- $\forall c \forall a : c \subseteq g \circ a(c) \text{ & } a \circ g(a) \leq a$



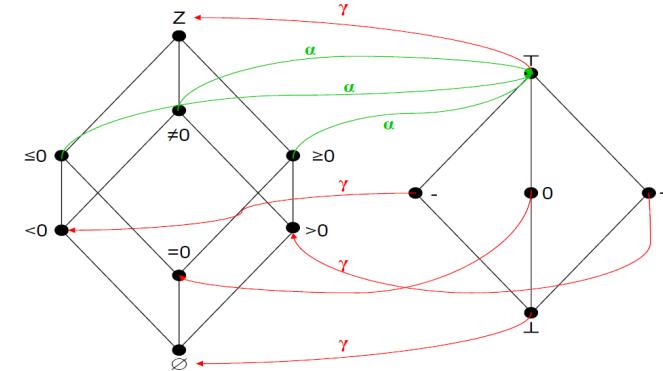
## Ejemplo

- $\forall c \forall a : a(c) \leq a \Leftrightarrow c \subseteq g(a)$

- $\forall c \forall a : c \subseteq g(a) \circ a(c) \& a \circ g(a) \leq a$



## Perdida de información



## Algunas propiedades

- Idempotencia
  - $a \circ g \circ a = a$
  - $g \circ a \circ g = g$
- Quasi inversas (podemos definir una en función de la otra)
  - $a(c) = \sqcap \{ a \mid c \subseteq g(a) \}$
  - $g(a) = \sqcup \{ c \mid a(c) \leq a \}$
- $a$  es el valor abstracto más preciso representando una propiedad concreta dada
- $g$  da la semántica de los valores abstractos en función de propiedades concretas

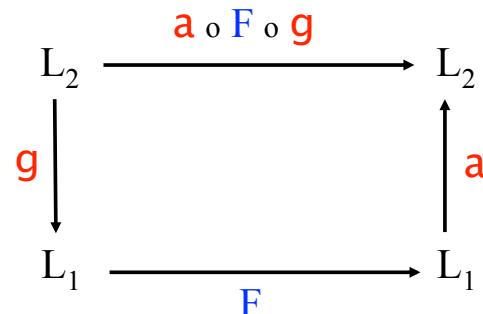
## Abstracción: Corrección vs. Completitud

- Corrección (usando  $Z, NZ, MZ$ )
  - Si no se cumple que  $Z \leq a(c)$
  - Sabemos que  $g(a(c))$  no incluye al 0
  - Por lo tanto podríamos dividir
- Incompletitud
  - Si se cumple que  $Z \leq a(c)$
  - No podemos concluir que  $c = 0$
  - Ya que  $g(a(c))$  incluye al 0 pero puede ser que  $c$  sea diferente de 0

## Abstracción de funciones

- Para toda función concreta  $F: C \rightarrow C$ , la correspondiente función abstracta  $F_a: A \rightarrow A$  puede ser determinada usando la conexión de Galois  $(a, g)$ 
  - $F_a(a) = a \circ F \circ g$
- Se lo llama "mejor abstracción posible de F".
- Como en general no calculamos  $(a, g)$ , se suele usar una aproximación de  $F_a$ 
  - $F_a(a) \leq F^{\#}(a)$

## Aproximación del punto fijo



Teorema:  $\text{lfp } F \subseteq g(\text{lfp } a \circ F \circ g)$

- Osea... Se puede aproximar el punto fijo concreto ( $F$ ) a partir del punto de su abstracción ( $F_a$ )

## Aproximación del punto fijo

- Dada una conexión de Galois entre  $A$  y  $C$ , se cumple la siguiente propiedad para toda  $F: C \rightarrow C$ 
  - $a(\text{lfp } F) \leq (\text{lfp } F_a)$ , o de manera equivalente
  - $\text{lfp } F \subseteq g(\text{lfp } F_a)$
- Luego, el cálculo sobre la abstracción es una sobreaproximación del cálculo en el dominio concreto!

## Abstracción de la collecting semantics

- Encontrar una conexión de Galois:
$$(\wp(S), \subseteq) \xrightarrow[\text{a}]{\text{g}} (S^{\#}, \leq)$$
- Encontrar una función:  $a \circ F \circ g \leq F^{\#}$

## Algebra con abstracciones

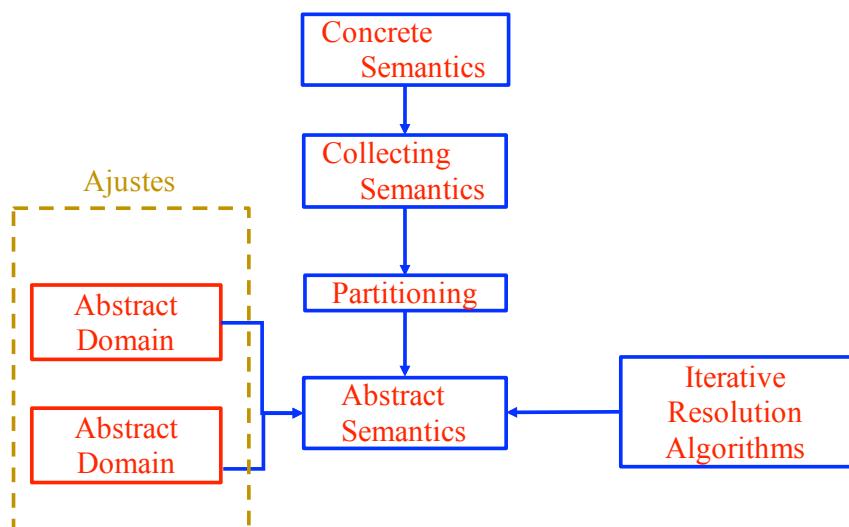
- Notación:  $E$  conjunto de todos los environments
- Conexión de Galois:

$$(\wp(E), \subseteq) \xrightleftharpoons[a]{g} (E^\#, \leq)$$

- $\cup, \cap$  es aproximado por  $\cup^\#, \cap^\#$
- Semántica  $\llbracket op \rrbracket$  aproximada usando  $\llbracket op \rrbracket^\#$

$$a \circ \llbracket op \rrbracket \circ g \subseteq \llbracket op \rrbracket^\#$$

## Metodología



## Ecuaciones en la semántica abstracta

```
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit
```

```
E1# = a ({n ⇒ ⊥})
E2# = ( [n = 0] # E1#) ∪# E4#
E3# = E2# ∩# a (]-∞, 999])
E4# = [n = n + 1] # E3#
E5# = E2# ∩# a ([1000, +∞])
```

## Dominios Abstractos

Environment:  $x \Rightarrow v, y \Rightarrow w, \dots$

Muchos tipos de aproximaciones

- Signos:  $x \Rightarrow +, y \Rightarrow 0, \dots$
- Intervalos:  
 $x \Rightarrow [a, b], y \Rightarrow [a', b'], \dots$
- Poliedros (relacional):  
 $x + y - 2z \leq 10, \dots$
- Matrices de diferencias (weakly relational):  
 $y - x \leq 5, z - y \leq 10, \dots$

## Ejemplo: signos

```

1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit

```

$$\begin{aligned}
E_1^{\#} &= \alpha(\{n \Rightarrow \perp\}) \\
E_2^{\#} &= ([n = 0] \# E_1^{\#}) \cup^{\#} E_4^{\#} \\
E_3^{\#} &= E_2^{\#} \cap^{\#} \alpha([-∞, 999]) \\
E_4^{\#} &= [n = n + 1] \# E_3^{\#} \\
E_5^{\#} &= E_2^{\#} \cap^{\#} \alpha([1000, +∞])
\end{aligned}$$

• Iteración 1

$$E_1^{\#} = \perp$$

$$E_2^{\#} = 0$$

$$E_3^{\#} = 0$$

$$E_4^{\#} = +$$

$$E_5^{\#} = 0 \cap + = \perp$$

• Iteración 2

$$E_1^{\#} = \perp$$

$$E_2^{\#} = \{0\} \cup \{+\} = T$$

$$E_3^{\#} = 0 \cap T = 0$$

$$E_4^{\#} = [+1] 0 = +$$

$$E_5^{\#} = 0 \cap + = \perp$$

$\perp$

$T$

$0$

$+$

## Problema

Como lidiar con reticulados de altura infinita?

**Solución:** Operadores de extrapolación

- **Widenning:** acelerar el proceso de iteración de Klenene, para llegar a un “post-fixpoint”
- **Narrowing:** para llegar a un punto fijo (no necesariamente el mínimo)

## Operador de widening

Reticulado  $(L, \leq)$ :  $\nabla : L \times L \rightarrow L$

- Operador de unión abstracta:

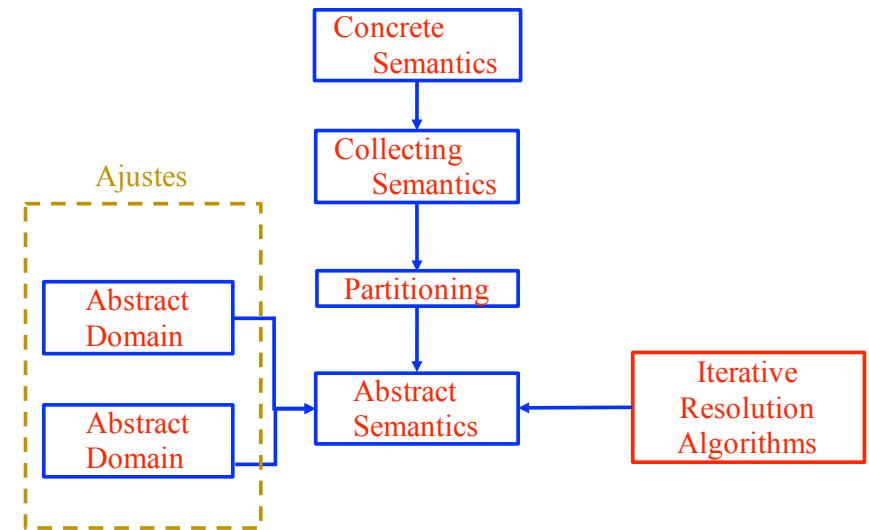
$$\forall x \forall y : x \sqcup y \leq x \nabla y$$

- Y debe forzar la convergencia: si  $x_0 \sqsubseteq x_1 \sqsubseteq \dots$

$$\begin{cases} y_0 = x_0 \\ y_{n+1} = y_n \nabla x_{n+1} \end{cases}$$

$(y_n)_{n \geq 0}$  es finita

## Metodología



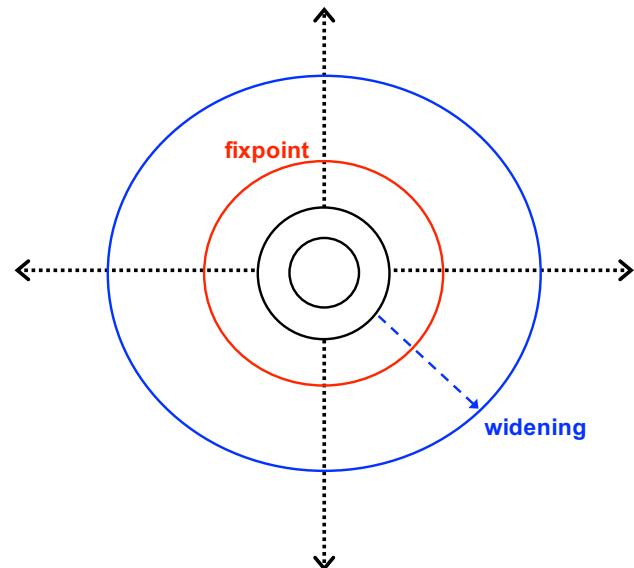
## Widening de intervalos

$$[a, b] \nabla [a', b']$$

- If  $a \leq a'$  then  $a$  else  $-\infty$
- If  $b' \leq b$  then  $b$  else  $+\infty$

→ Se puede pasar del punto

## Widening y punto fijo



## Imprecisión por widening

- ```
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit; t[n] = 0; // t has 1500 elements
```
- $E_5^{\#} = [1000, +\infty[$
- La información esta presente en las ecuaciones
- Falso positivo!!!

## Iterando con widening

```
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit
```

$$(E_2^{\#})_{n+1} = (E_2^{\#})_n \nabla ([n = 0]^{\#} (E_1^{\#})_n \cup^{\#} (E_4^{\#})_n)$$

Iteración 1 (union):  $E_2^{\#} = [0, 0]$

Iteración 2 (union):  $E_2^{\#} = [0, 1]$

Iteración 3 (widening):  $E_2^{\#} = [0, +\infty] \Rightarrow$  estable

## Operador Narrowing

- Mejora el resultado del widening
- Reticulado ( $L, \leq$ ):  $\Delta : L \times L \rightarrow L$
- Operador de intersección abstracta:
- $$\forall x \forall y : x \leq y \Rightarrow x \leq x \Delta y \leq y \quad (x \sqcap y \leq x \Delta y)$$
- Debe forzar convergencia: si  $x_0 \geq x_1 \geq \dots$  (decrecientes)

$$\begin{cases} y_0 = x_0 \\ y_{n+1} = y_n \Delta x_{n+1} \end{cases}$$

$(y_n)_{n \geq 0}$  es finita

## Narrow en intervalos

$$[a, b] \Delta [a', b']$$

- If  $a = -\infty$  then  $a'$  else  $a$
- If  $b = +\infty$  then  $b'$  else  $b$

→ Refina las cotas que quedaron abiertas

## Iteración con narrowing

```

1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit; t[n] = 0;

```

$$\begin{aligned}
 E_1^{\#} &= f \{n \Rightarrow \Omega\} \\
 E_2^{\#} &= ([n = 0]^{\#} E_1^{\#}) \cup^f E_4^{\#} \\
 E_3^{\#} &= E_2^{\#} \cap^f f[-\infty, 999] \\
 E_4^{\#} &= [n = n + 1]^{\#} E_3^{\#} \\
 E_5^{\#} &= E_2^{\#} \cap^f f[1000, \infty]
 \end{aligned}$$

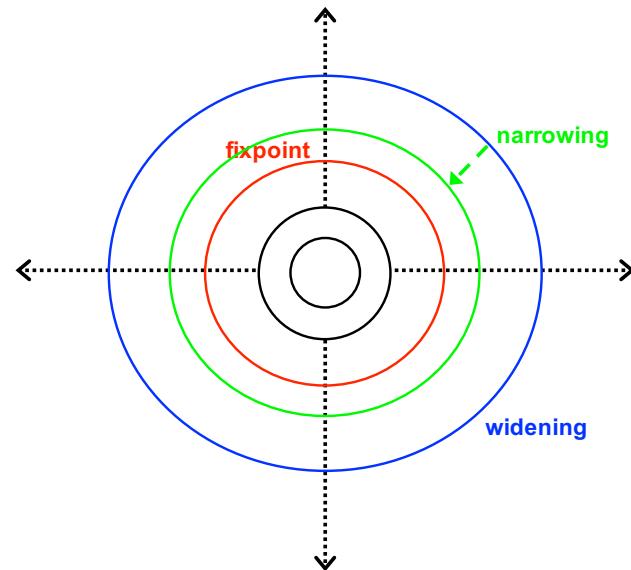
$$(E_2^{\#})_{n+1} = (E_2^{\#})_n \Delta \left( \underbrace{[n = 0]}_{[1, 1000]} \# (E_1^{\#})_n \cup^{\#} (E_4^{\#})_n \right)$$

Inicio de la iteración:  $E_2^{\#} = [0, +\infty[$

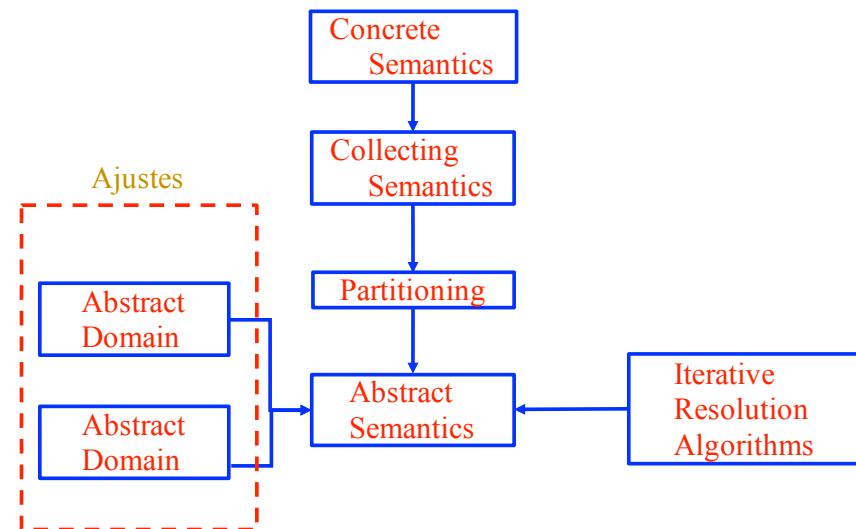
Iteración 1:  $E_2^{\#} = [0, 1000] \Rightarrow$  stable

Consecuencia:  $E_5^{\#} = [1000, 1000]$

## Narrowing y puntos fijos



## Metodología



## Seleccionando dominios abstractos

```
1: n = 0;
2: k = 0;
3: while n < 1000 do
4:   n = n + 1;
5:   k = k + 1;
6: end
7: exit
```

- Usando intervalos:

$$E_4^{\#} = \langle n \Rightarrow [0, 1000], k \Rightarrow [0, +\infty[ \rangle$$

- Usando Poliedros o DBMs:

$$E_4^{\#} = \langle 0 \leq n \leq 1000, 0 \leq k \leq 1000, n - k = 0 \rangle$$

## Bibliografia

- Mirar slides de Patrick Cousot
- Un curso de David Pichardie
  - <http://limerick.cost-ico701.org/home/abstract-interpretation>
- [Principles of Program Analysis](#). Flemming Nielson, Hanne Riis Nielson, Chris Hankin.