

Análisis Automático de Programas

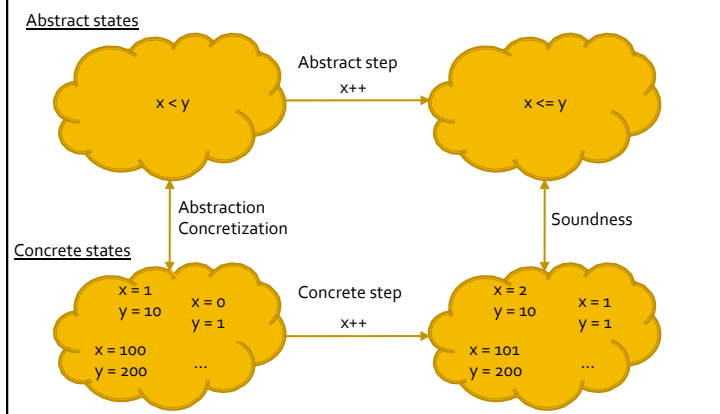
Interpretación abstracta

Diego Garbervetsky, Guido de Caso
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 (on 1 slide)



Interpretación abstracta

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

Aproximación

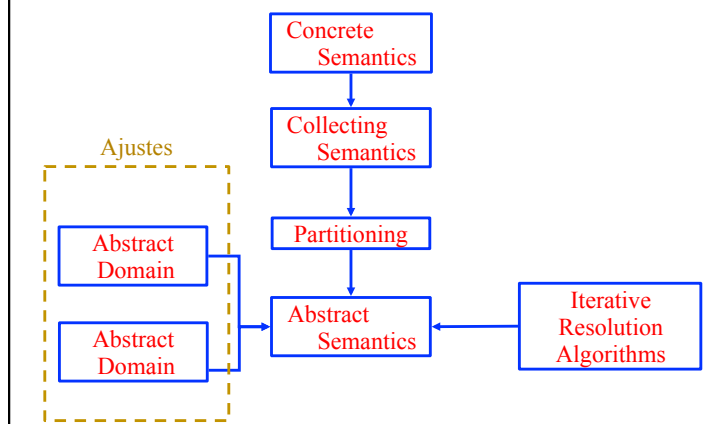
La idea principal del framework de Interpretación Abstracta es la de formalizar la noción de **aproximación**

- 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

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 instancias de los esquemas para ver cual es el que mejor ajusta

Metodología



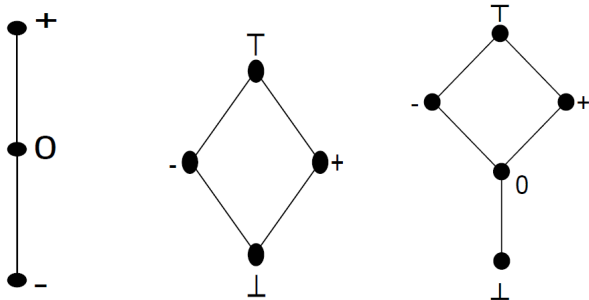
Reticulados y puntos fijos

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

Reticulados

$(\mathbb{Z} \cup \{-\infty, \infty\}, \leq, -\infty, \infty, \max, \min)$

$(\wp(\mathbb{Z}), \subseteq, \emptyset, \mathbb{Z}, \cup, \cap)$



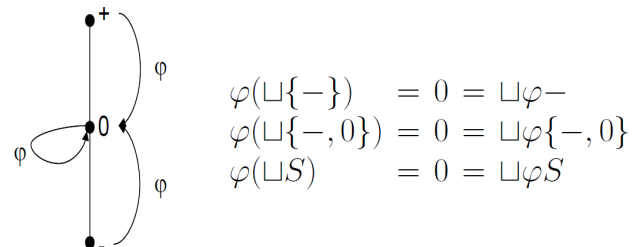
Puntos fijos

- 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.
 - $gfp F = \sqcup FP_F$

Propiedades

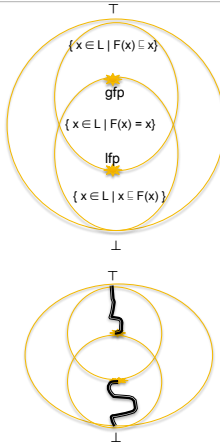
- $F: L \rightarrow L$ es **monotona** si
 - $\forall x, y \in L, x \sqsubseteq y \rightarrow F(x) \sqsubseteq F(y)$
- F es **upper continua** (preserva supremo) si
 - $\forall X = \{x_1 \sqsubseteq \dots \sqsubseteq x_n\} F(\sqcup X) = \sqcup F(X)$
 - $\sqcup F(X)$ representa $F(x_1) \sqcup \dots \sqcup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

Ejemplo



Teoremas del punto fijo

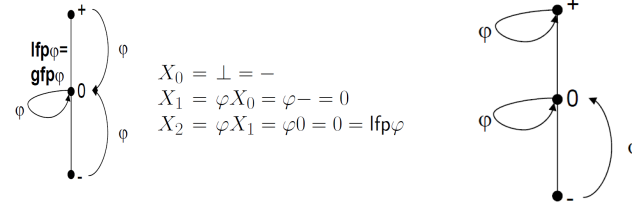
- Teorema de Knaster-Tarski: Si $F: L \rightarrow L$ es monótona y L es un reticulado completo, el conjunto FP_F de puntos fijos de F es también un reticulado completo y $lfp = \sqcap FP_F$



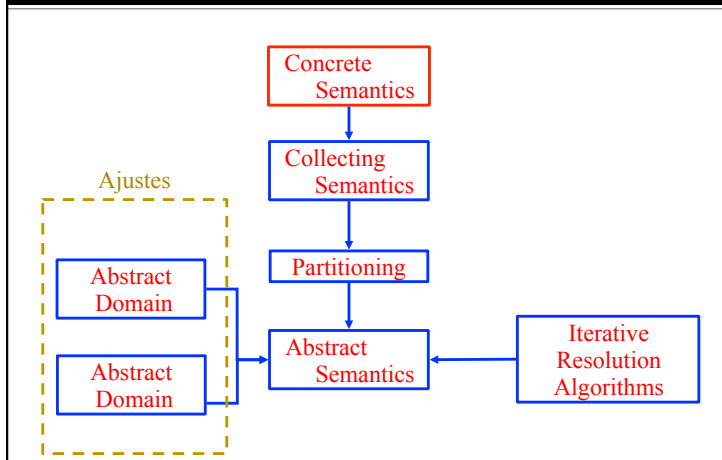
- Teorema de Kleene: Si $F: L \rightarrow L$ es continua, L es un reticulado completo entonces $lfp F$ se define como:

$$\sqcup \begin{cases} F_0 & = \perp \\ F_{n+1} & = F(F_n) \end{cases}$$

Ejemplo



Metodología



Semantica Concreta

Semántica operacional small-step: (Σ, \rightarrow)

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

$$s \rightarrow s'$$

Ejemplo:

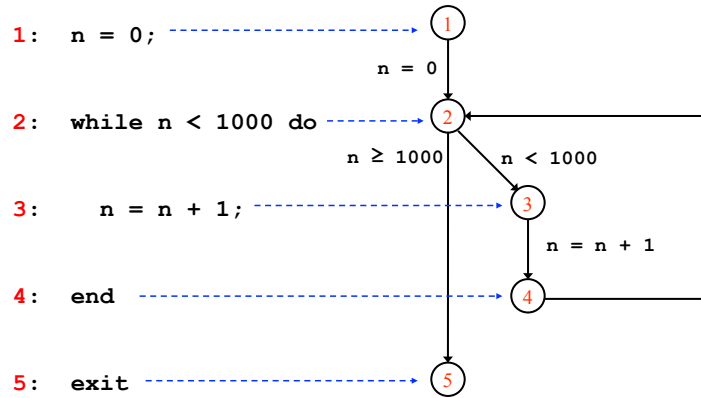
```

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

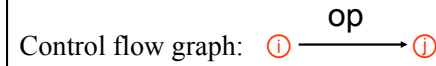
$\langle 1, n \Rightarrow \perp \rangle \rightarrow \langle 2, n \Rightarrow 0 \rangle \rightarrow \langle 3, n \Rightarrow 0 \rangle \rightarrow \langle 4, n \Rightarrow 1 \rangle$
 $\rightarrow \langle 2, n \Rightarrow 1 \rangle \rightarrow \dots \rightarrow \langle 5, n \Rightarrow 1000 \rangle$

Valor indefinido concreto

Control Flow Graph



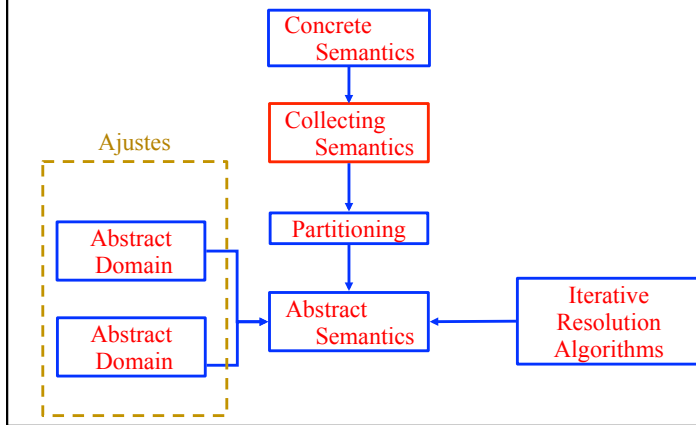
Relación de transición



Semántica operacional: $\langle \textcircled{i}, \epsilon \rangle \rightarrow \langle \textcircled{i}, \llbracket op \rrbracket \epsilon \rangle$

Semántica de op

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

Qué recolectamos?

Depende del análisis

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

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: $F : (\emptyset(\Sigma), \subseteq) \rightarrow (\emptyset(\Sigma), \subseteq)$

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

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

Ejemplo

```

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

```

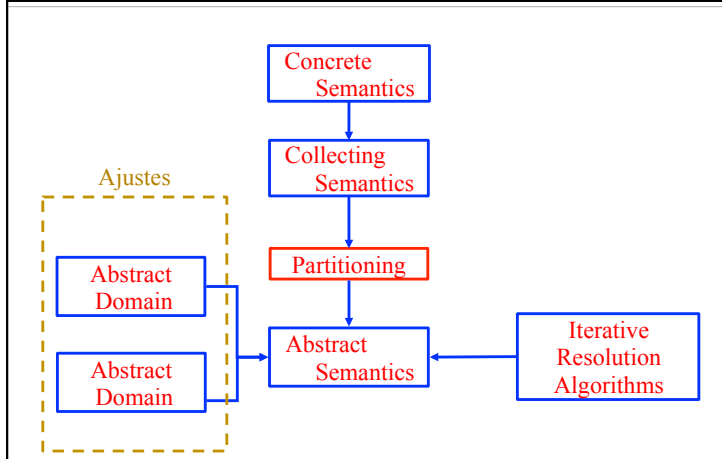
$$S = \{\langle 1, n \Rightarrow \perp \rangle, \langle 2, n \Rightarrow 0 \rangle, \langle 3, n \Rightarrow 0 \rangle, \langle 4, n \Rightarrow 1 \rangle, \langle 2, n \Rightarrow 1 \rangle, \dots, \langle 5, n \Rightarrow 1000 \rangle\}$$

Cálculo del funcional

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

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

Metodología



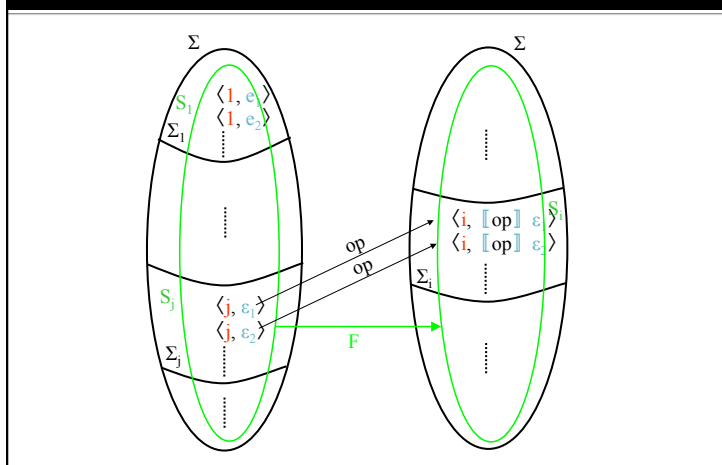
Partitioning

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

- Objetivo: Colectar información en cada punto del programa

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

Idea



Ecuaciones de semántica

- Notación: E_i = conjunto de valuaciones en el punto del programa i
 - Proyectar 2^{da} componente de $\{ \langle i, \epsilon \rangle \}$
- Sistema de ecuaciones de semántica :

$$E_i = U \{ [op] E_j \mid \textcircled{i} \xrightarrow{op} \textcircled{i} \in CFG(P) \}$$

- Solución del sistema: $S = \text{lfp } F$
 - Recordar que podemos calcular S como
 - $S^0 = (\perp, \dots, \perp)$, $S^{n+1} = F(S_1^n) = (S_1^n, \dots, S_n^n)$
 - Usando E
 - $E^0 = (\perp, \dots, \perp)$, $E^{n+1} = U \{ [op] E_j^n \mid \textcircled{i} \xrightarrow{op} \textcircled{i} \in CFG(P) \}$

Ejemplo

```

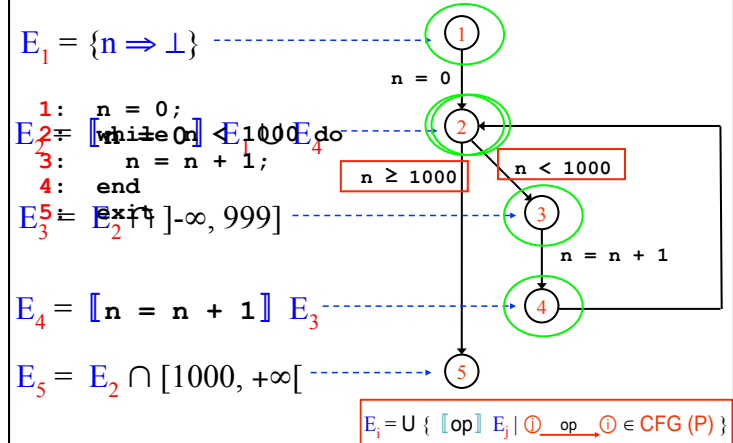
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit

```

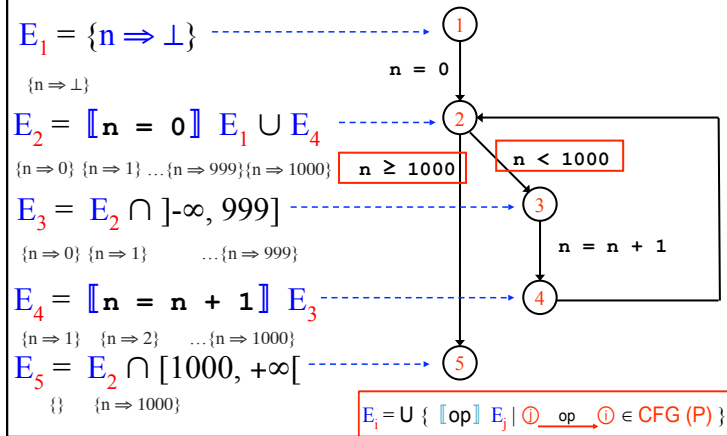
$E_i = U \{ \llbracket op \rrbracket E_j \mid \langle _op_ \rangle \in CFG(P) \}$

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

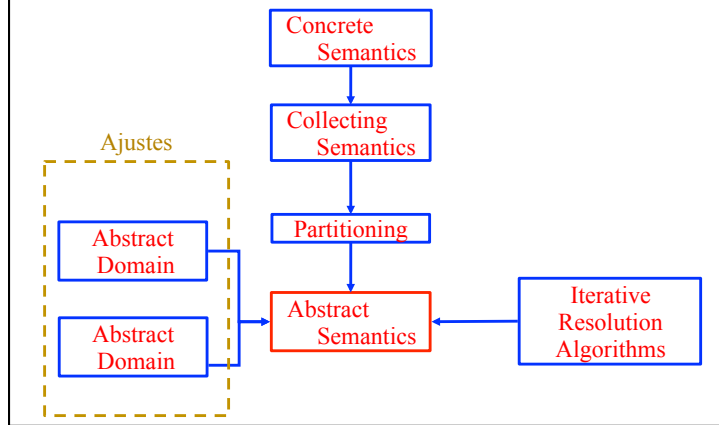
Ejemplo



Ejemplo



Metodología



Aproximación

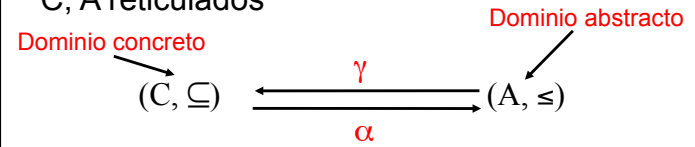
Problema: Calcular una aproximación sound de $S: S^\#$

$$S \subseteq S^\#$$

Solución: conexiones de Galois

Conexiones de Galois

C, A reticulados

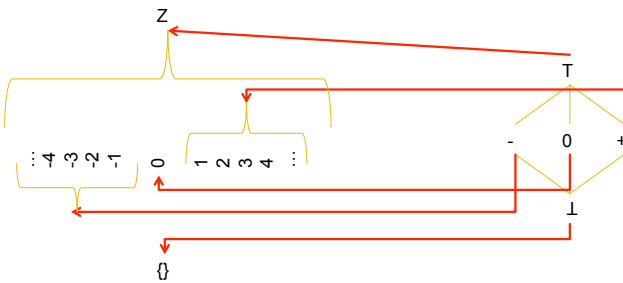


- $\forall c \forall a : \alpha(c) \leq a \Leftrightarrow c \subseteq \gamma(a)$
- $\forall c \forall a : c \subseteq \gamma \circ \alpha(c) \ \& \ \alpha \circ \gamma(a) \leq a$

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

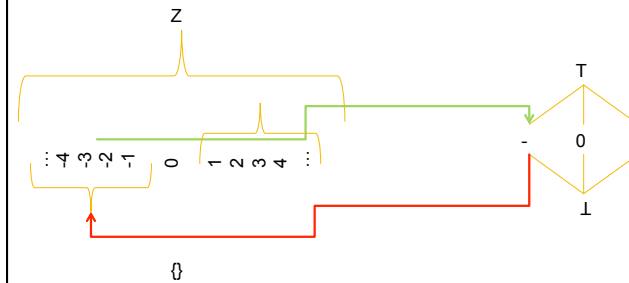
Ejemplo

- $\forall c \forall a : \alpha(c) \leq a \Leftrightarrow c \subseteq \gamma(a)$
- $\forall c \forall a : c \subseteq \gamma \circ \alpha(c) \ \& \ \alpha \circ \gamma(a) \leq a$

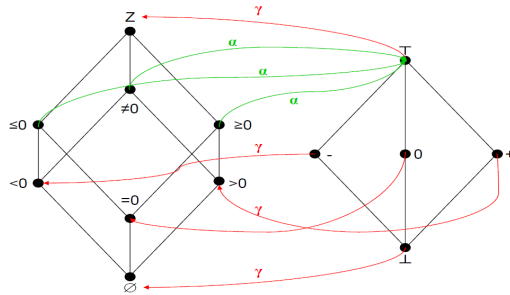


Ejemplo

- $\forall c \forall a : \alpha(c) \leq a \Leftrightarrow c \subseteq \gamma(a)$
- $\forall c \forall a : c \subseteq \gamma \circ \alpha(c) \ \& \ \alpha \circ \gamma(a) \leq a$



Perdida de información



Algunas propiedades

- Idempotencia
 - $\alpha\gamma\alpha = \alpha$
 - $\gamma\alpha\gamma = \gamma$
- Quasi inversas (podemos definir una en función de la otra)
 - $\alpha(c) = \sqcap \{ a \mid c \subseteq \gamma(a) \}$
 - $\gamma(a) = \sqcup \{ c \mid \alpha(c) \leq a \}$
 - α es el valor abstracto más preciso representando una propiedad concreta dada
 - γ 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 $Z \leq \alpha(c)$
 - Sabemos que $\gamma(\alpha(c))$ no incluye al 0
 - Por lo tanto podríamos dividir
- Incompletitud
 - Si se cumple que $Z \leq \alpha(c)$
 - No podemos concluir que $c = 0$
 - Ya que $\gamma(\alpha(c))$ incluye al 0 pero puede ser que c mismo no lo sea

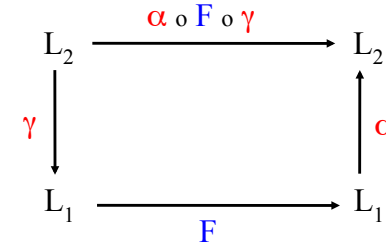
Abstracción de funciones

- 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 (α, γ)
 - $F_a(a) = \alpha \circ F \circ \gamma$
- Se lo llama "mejor abstracción posible de F".
- Como en general no calculamos (α, γ) , se suele usar una aproximación de F_a
 - $F_a(a) \leq F\#(a)$

Aproximación del punto fijo

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

Aproximación del punto fijo



Teorema: $\text{lfp } F \subseteq \gamma(\text{lfp } \alpha \circ F \circ \gamma)$

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

Abstracción de la collecting semantics

- Encontrar una conexión de Galois:

$$(\emptyset(\Sigma), \subseteq) \begin{matrix} \xleftarrow{\gamma} \\ \xrightarrow{\alpha} \end{matrix} (\Sigma^\#, \leq)$$

- Encontrar una función: $\alpha \circ F \circ \gamma \leq F^\#$

Algebra con abstracciones

- Notacion: E conjunto de todos los environments
- Conexión de Galois:

$$(\emptyset(E), \subseteq) \begin{matrix} \xleftarrow{\gamma} \\ \xrightarrow{\alpha} \end{matrix} (E^\#, \leq)$$

- \cup, \cap es aproximado por $\cup^\#, \cap^\#$
- Semántica $[[op]]$ aproximada usando $[[op]]^\#$

$$\alpha \circ [[op]] \circ \gamma \subseteq [[op]]^\#$$

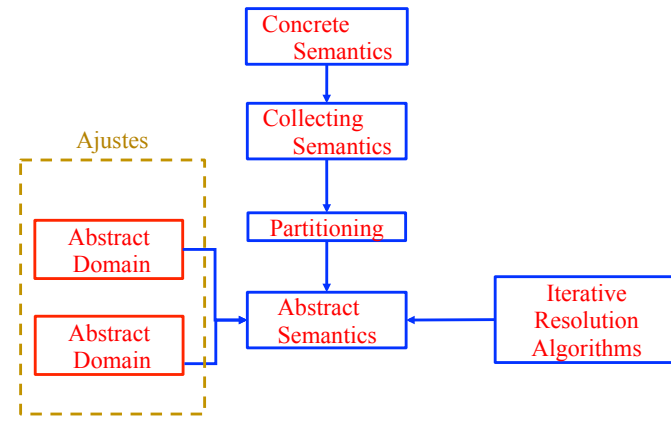
Ecuaciones en la semantica abstracta

```

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

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

Metodología



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):

$$y - x \leq 5, z - y \leq 10, \dots$$

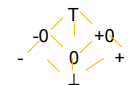
Ejemplo: signos

```

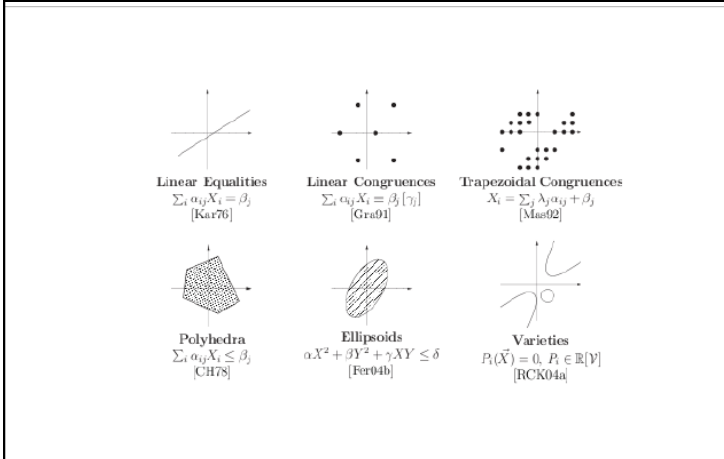
1: n = 0;
2: while n < 1000 do
3:   n = n + 1;
4: end
5: exit
    
```

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

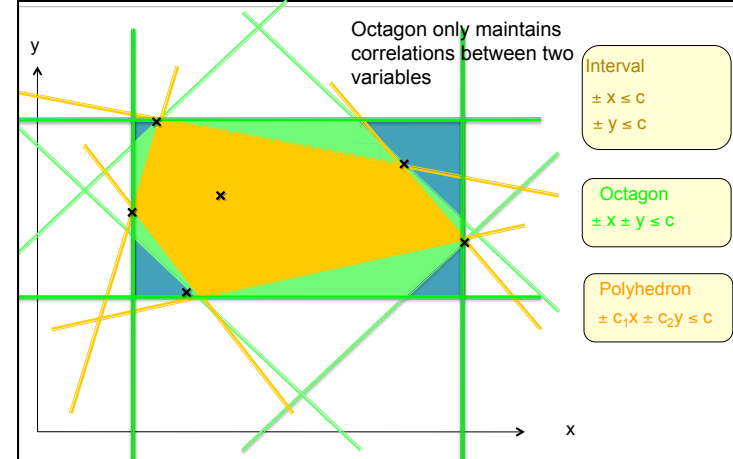
Iteración 1	Iteración 2	Iteración 3	Iteración 4	
$\bullet E_1^\# = \perp$	$\bullet E_1^\# = \perp$	$\bullet E_1^\# = \perp$	$\bullet E_1^\# = \perp$	$\bullet E_1^\# = \perp$
$\bullet E_2^\# = 0$	$\bullet E_2^\# = \{0\} \cup \{+\} = T$	$\bullet E_2^\# = \{0\} \cup \{+\} = T$	$\bullet E_2^\# = \{0\} \cup \{+\} = T$	$\bullet E_2^\# = 0+$
$\bullet E_3^\# = 0$	$\bullet E_3^\# = 0 \cap T = 0$	$\bullet E_3^\# = T \cap T = T$	$\bullet E_3^\# = T \cap T = T$	$\bullet E_3^\# = 0+$
$\bullet E_4^\# = +$	$\bullet E_4^\# = [+1] \ 0 = +$	$\bullet E_4^\# = [+1] \ 0 = +$	$\bullet E_4^\# = [+1] \ T = T$	$\bullet E_4^\# = +$
$\bullet E_5^\# = +$	$\bullet E_5^\# = 0 \cap + = T$	$\bullet E_5^\# = T \cap + = +$	$\bullet E_5^\# = T \cap + = +$	$\bullet E_5^\# = +$



Ejemplos de reticulados



Numerical Abstractions



Ejemplo: intervalos

```

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

- Iteration 1: $E_2^\# = [0, 0]$
- Iteration 2: $E_2^\# = [0, 1]$
- Iteration 3: $E_2^\# = [0, 2]$
- Iteration 4: $E_2^\# = [0, 3]$
- ...

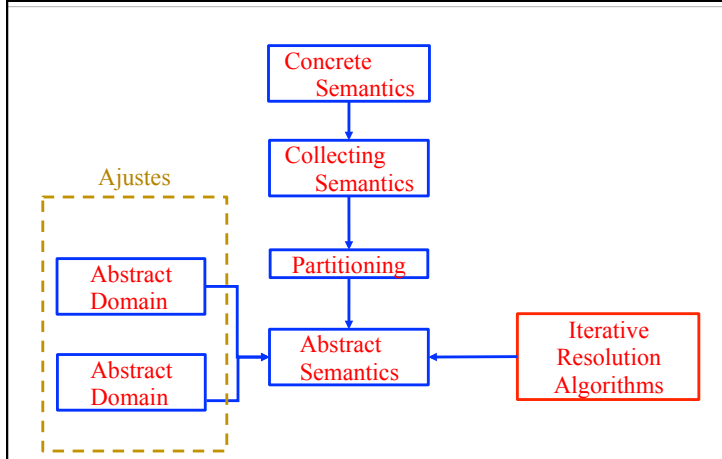
Problema

Como lidiar con reticulados de altura infinita?

Solución: Operadores de extrapolación

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

Metodología



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

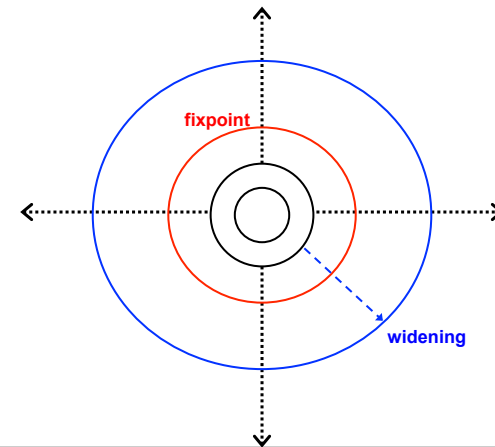
Widening de intervalos

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

- 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



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

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

```

False positivo!!!

- $E_5^\# = [1000, +\infty[$

- La información esta presente en las ecuaciones

Operador Narrowing

- Mejora el resultado del widening

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

- Operador de intersección abstracta:

$$\forall x \forall y : x \leq y \Rightarrow x \leq x \Delta y \leq y \quad (x \cap y \leq x \Delta y)$$

- 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

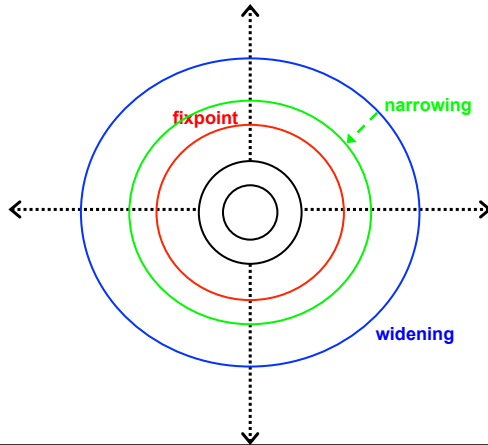
Narrow en intervalos

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

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

↪ Refina las cotas que quedaron abiertas

Narrowing y puntos fijos



Iteracion con narrowing

```

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

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

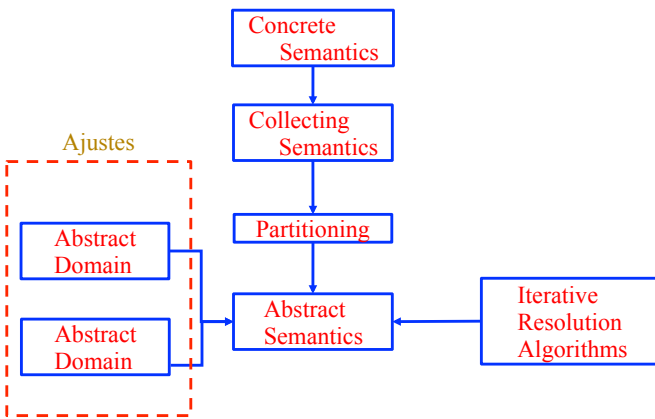
$$(E_2^\#)_{n+1} = (E_2^\#)_n \Delta \left(\underbrace{[n = 0] \# (E_1^\#)_n \cup (E_4^\#)_n}_{[1, 1000]} \right)$$

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

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

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

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:

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

Bibliografía

- 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.