# Solución de Problemas con CCP Propagadores

slides basados en el curso “constraint Programming” de Christian Schulte [2]
Profesor: Camilo Rueda [1]

[1] Universidad Javeriana-Cali,
[2] KTH Royal Institute of Technology, Sweden

PUJ 2008

# Problemas de satisfacción de restricciones

$$\boxed{CSP = \langle V, U, R \rangle}$$

- Un conjunto $V$ de variables
  $V = \{x_1, x_2, ..., x_n\}$
- Un universo $U$ de valores posibles (tipo)
  todas las variables toman valores de $U$
- Un conjunto $R$ de restricciones
  - cada restricción involucra algunas de las variables
  - establecen valores aceptables (sus soluciones)

# Componentes de las restricciones

una restricción $r$ define:

- sus variables
  $$var(r) = (x_1, ..., x_k) \in V^k$$
- sus soluciones
  $$sol(r) \subseteq U^k$$

# Asignaciones

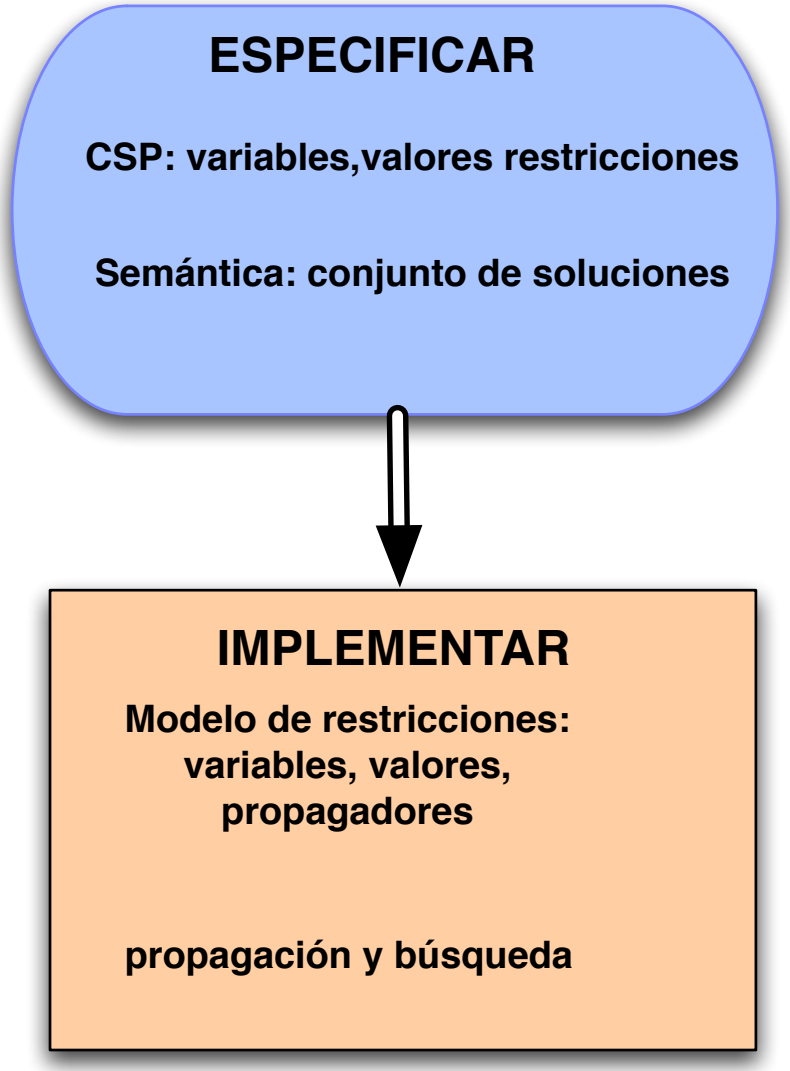
- una asignación a define qué valor toma cada variable:
  $a : V \rightarrow U$
- ua asignación a es solución de una restricción r (se escribe $a \in r$) ssi
  - $var(r) = (x_1, ..., x_k)$
  - $(a(x_1), ..., a(x_k)) \in sol(r)$

# Solución de un CSP

- una asignación $a \in V \rightarrow U$ es una solución de un CSP $P = \langle V, U, R \rangle$ si
  - $a \in r$ para toda $r \in R$
- las soluciones $sol(P)$ de $P$ se definen como

$$sol(P) = \{s : V \rightarrow U \mid s \text{ es solución de } P\}$$

# Especificar vs implementar



# Especificar vs implementar (2)

- Un modelo implementa un CSP
  - cuando tienen el mismo conjunto de soluciones

# Resolver un CSP

- Encontrar una implementación: el modelo de restricciones
- El modelo usa propagadores
- Los propagadores deben cumplir ciertas propiedades

# Modelo de restricciones

- Ofrece una implementación de un CSP
  - asegurarse de que efectivamente implementa el CSP
- en lugar de restricciones se tienen propagadores

# Store de restricciones

- función que asocia variables con conjuntos de valores:
  $s \in V \to \mathbb{P}(U)$
  - un conjunto de stores se denota $S = V \to \mathbb{P}(U)$
- Los propagadores y ramificadores operan sobre stores

# Fortaleza de un store

- un store $s_1$ es más fuerte que un store $s_2$ (escrito $s_1 \leq s_2$) si y solo si
  $$s_1(x) \subseteq s_2(x) \quad \text{para toda } x \in V$$
- $s_1$ es estrictamente más fuerte que $s_2$ (escrito $s_1 < s_2$) ssi
  $$s_1 \leq s_2 \wedge s_1 \neq s_2$$

# Propiedades de los propagadores

- contractor: debe calcular stores más fuertes
- propagador: función de stores a stores, $p \in S \rightarrow S$
  - $p(s) \leq s$ para todo store $s$
- si $p$ implementa la restricción $r$,
  - $p$ nunca elimina una solución de $r$

# Propiedades de los propagadores (2)

- Escribimos $a \in s$, para una asignación $a$ y store $s$ si
  - $a(x) \in s(x)$ para toda variable $x \in V$
- Definimos
  - $store_a(x) = \{a(x)\}$
    ( $store_a$ es la asignación $a$ vista como store )
  - propiedad: $a \in s \Leftrightarrow store_a \leq s$

# Implementar restricción

- el propagador $p$ implementa la restricción $c$ si

$$\boxed{a \in c \text{ si y solo si } p(store_a) = store_a}$$

  o sea,
  - las soluciones de $c$ son puntos fijos de $p$
- Lo anterior no basta

# los propagadores deben ser monotónicos

$p \in S \to S$ es

- contractor $p(s) \leq s$
- monotónico $s_1 \leq s_2 \Rightarrow p(s_1) \leq p(s_2)$

$$a \in s \Leftrightarrow store_a \leq s$$
$$\Rightarrow p(store_a) \leq p(s)$$
monotonicidad
$$\Leftrightarrow store_a \leq p(s)$$
$$\Leftrightarrow a \in p(s)$$

# asignaciones no válidas

- suponga que $p$ implementa c, que $a \notin c$ y sea $s = store_a$. Entonces $p$ falla en $s$

$$
\begin{aligned}
& a \notin s \Leftrightarrow p(s) \neq s \\
& \quad \Rightarrow p(s) < s \\
& \qquad \text{(contractor)} \\
& \quad \Rightarrow \exists x \in V \text{ tal que } p(s)(x) \subset s(x) \\
& \quad \Rightarrow \exists x \in V \text{ tal que } p(s)(x) = \emptyset \\
& \quad \Rightarrow p \text{ falla en } s
\end{aligned}
$$

# Modelo de restricciones

- un modelo de restricciones $M = \langle V, U, P \rangle$ se define mediante
  - un conjunto de variables $V$
  - un conjubto de valores $U$
  - un conjunto de propagadores $P$

# Soluciones de un propagador

- Las soluciones $sol(p)$ de un propagador $p$ se definen como $\{a \in V \to U \mid store_a = p(store_a)\}$
- Las soluciones $sol(M)$ de un modelo de restricciones $M = \langle V, U, P \rangle$ se definen como $\{a \in V \to U \mid a \in sol(p) \text{ para todo } p \in P\}$

| Un modelo de restricciones $M = \langle V, U, P \rangle$ implementa un CSP $C$ si |
| --- |
| $sol(M) = sol(C)$ |

# Soluciones (2)

- Interesan las soluciones a un modelo $M$ obtenidas a partir de un store $s$:

  $$sol(M, s)$$

  definidas como

$$\boxed{sol(M, s) = \{a \in sol(M) \mid a \in s\}}$$

| Propiedades de un propagador |
| --- |
| para todo modelo $M = \langle V, U, P \rangle$ y store $s$ un propagador $p$ debe cumplir $sol(M, s) = sol(M, p(s))$ |
| los propagadores preservan las soluciones |

# El resultado calculado

- supongamos $propagar(\langle V, U, P\rangle, s) = s_r$

  $sol(\langle V, U, P\rangle, s) = sol(\langle V, U, P\rangle, s_r)$
  (no se eliminan soluciones)

  para todo $p \in P : p(s_r) = s$
  (no hay más propagación posible)
  (el punto fijo simultáneo más grande)

# El resultado calculado (2)

- supongamos $propagar(\langle V, U, P\rangle, s) = s_r$, entonces

| $s_r$ es el máximo punto fijo simultáneo con $s_r \leq s$ |
| --- |
| es decir, para todo $p \in P$:<br>$p(s_r) = s$ |

# Un propagador mejorado

- Idea: un propagador poda dominios de algunas pocas variables
  - repropagar solamente propagadores que comparten alguna de ellas
- mantener un conjunto de propagadores “activos”
  - no se sabe si ya llegaron a su punto fijo en el store actual
  - todos los demás ya lo hicieron
  - propagar solamente los “activos”

# Variables de un propagador

- Variables $var(p)$ del propagador $p$
  - variables de interés para $p$
- No se consideran entradas de otras variables
  para todo $s_1, s_2 \in S$
  si ocurre que
  $\forall x \in var(p) : s_1(x) = s_2(x)$
  entonces, $\forall x \in var(p)$ :
  $$p(s_1)(x) = p(s_2)(x) \ )$$
- No se afectan otras variables con los resultados:
  para todo $s \in S$ y para toda $x \in (V - var(p))$
  $$p(s)(x) = s(x)$$

# Propagador mejorado

$propague((V, U, P), s_0)$
$s := s_0;\ A := P;$
*while* $A \neq \emptyset$ *do*
*escoja* $p \in A$;
$s' := p(s);\ A := A - \{p\};$
$VM := \{x \in V \mid s(x) \neq s'(x)\};$
$PA := \{q \in P \mid \exists x \in var(q) : x \in VM\};$
$A := A \cup PA;$
$s := s';$
*return* $s$;

# Propiedades

- Qué calcula
  - puntos fijos simultáneos
  - el punto fijo más grande
  - prueba: definir invariante del ciclo
- Terminación?
  - los stores ya no son estrictamente más fuertes

# Invariante

- El ciclo mantiene la propiedad
  - para todo $p \in P - A \Rightarrow p(s) = s$
  - después de terminación ($A = \emptyset$):
    para todo $p \in P \Rightarrow p(s) = s$

# Mejorar aun más el propagador

- Idea: conocer sobre el punto fijo de un propagador
  - el propagador anterior tiene solo conocimiento implícito
  - se debe hacer explícito:

cada propagador provee información sobre el punto fijo

# Mejorar aun más el propagador: ejemplo

- Suponga el propagador

$$p(s) = \{x \mapsto (s(x) \cap \{1, 2, 3\}\}$$

  - implementa la restricción de dominio $x \in \{1, 2, 3\}$
- después de ejecutar $p$ una vez, ya no hay que volver a hacerlo:

  si $s' \leq s$ entonces $p(s') = s'$
- se puede eliminar $p$ del modelo

# Propagadores redundantes

- Un store $s$ vuelve redundante un propagador $p$ ssi $\forall s' \leq s : p(s') = s$
  - todo store más fuerte es punto fijo
  - $p$ se deduce de $s$
  - es decir, $s$ vuelve redundante el propagador $p$

# Considerar el propagador de $\leq$

| $V = \{x, y\}, \quad U = \{0, ..., 5\}$ |
| --- |
| *propagador* $p_{\leq}$ *para* $x \leq y$ : <br> $p_{\leq}(s) = \{x \mapsto \{d \in s(x) \mid d \leq max(s(y))\}$ <br> $y \mapsto \{d \in s(y) \mid d \geq min(s(x))\}\}$ |

# Considerar el propagador de $\leq$ (2)

- Después de ejecutar $p_{\leq}$ sbre un store $s$ tenemos
  $$p_{\leq}(p_{\leq}(s)) = p_{\leq}(s)$$
  - $max(s(y))$ no cambia
  - $min(s(x))$ no cambia
- pero: como $va(p_{\leq}) = \{x, y\}$, entonces $p_{\leq}$ se agrega a $PA$
  - esto es innecesario!

# Nociones: idempotencia

- Una función $f \in X \rightarrow X$ es idempotente si $\forall x \in X : f(f(x)) = f(x)$
- propiedad demasiado fuerte para un propagador: debe cumplirse para todo store

# Nociones: idempotencia débil

- Una función $f \in X \to X$ es idempotente en $x \in X$ si $f(f(x)) = f(x)$
  - propiedad sobre un solo elemento
- si un propagador $p$ es idempotente en $s$, no necesariamente es idempotente en $s' \leq s$

# Cómo calcular idempotencia?

- dado un store $s$ y un propagador $p$
  - el store $s$ vuelve redundante a $p$?
    ensayar todo $s' \leq s$: demasiado costoso
  - es $p$ idempotente en $s$?
    aplicar $p$ a $s$? eso es lo que se trata de evitar!

# Cómo calcular idempotencia?

- dado un store $s$ y un propagador $p$
  - el store $s$ vuelve redundante a $p$?
    ensayar todo $s' \leq s$: demasiado costoso
  - es $p$ idempotente en $s$?
    aplicar $p$ a $s$? eso es lo que se trata de evitar!

Solución:

- el propagador retorna un estado, además del resultado:
  $p$ es una función $p \in S \rightarrow SM \times S$
  donde
  $SM = \{fijo, nofijo, redundante\}$

# Propagador con estado

- sea un propagador $p$ y un store $s$

| resultado | significado |
|---|---|
| $p(s) = (fijo, s')$ | $s'$ es punto fijo de $p$ |
| $p(s) = (redundante, s')$ | $s'$ vuelve redundante a $p$ |
| $p(s) = (nofijo, s')$ | no se sabe nada |

# Propagador con estado

- sea un propagador $p$ y un store $s$

| resultado | significado |
|---|---|
| $p(s) = (fijo, s')$ | $s'$ es punto fijo de $p$ |
| $p(s) = (redundante, s')$ | $s'$ vuelve redundante a $p$ |
| $p(s) = (nofijo, s')$ | no se sabe nada |

Solución:

- el propagador retorna un estado, además del resultado:
  $p$ es una función $p \in S \rightarrow SM \times S$
  donde
  $SM = \{fijo, nofijo, redundante\}$

# Propagador de $\leq$ con estado

*propagador* $p_{\leq}$ *para* $x \leq y$ :

$p_{\leq}(s) =$
  **si** $max(s(x)) \leq min(s(y))$ **entonces**
    $(redundante, s)$
  **sino**
    $(fijo,$
     $\{x \mapsto \{d \in s(x) \mid d \leq max(s(y))\}$
      $y \mapsto \{d \in s(y) \mid d \geq min(s(x))\}\}$ $)$

# Propagador mejorado aun más

$propague((V, U, P), s_0)$
$s := s_0;\ A := P;$
*while* $A \neq \emptyset$ *do*
*escoja* $p \in A$;
$(e, s' :) = p(s);\ A := A - \{p\};$
*if* $e =$ *redundante then* $P := P - \{p\}$ *end*
$VM := \{x \in V \mid s(x) \neq s'(x)\};$
$PA := \{q \in P \mid \exists x \in var(q) : x \in VM\};$
*if* $e =$ *fijo then* $PA := PA - \{p\}$ *end*
$A := A \cup PA;$
$s := s';$
*return* $(P, s);$

# Eventos de propagación

- Para muchos propagadores
  - es simple decidir si todavía se está en punto fijo para un dominio que cambió
  - con base en cómo cambió el dominio
- La descripción sobre cómo cambia el dominio la hace un

evento de propagación

# El propagador de $\leq$

| *propagador* $p_{\leq}$ *para* $x \leq y$ : |
| --- |
| $p_{\leq}(s) = \{x \mapsto \{d \in s(x) \mid d \leq max(s(y))\}$ <br> $y \mapsto \{d \in s(y) \mid d \geq min(s(x))\}\}$ |

debe propagarse solamente cuando
$min(s(x))$ o $max(s(y))$ cambie

# El propagador de $\neq$

| *propagador* $p_{\neq}$ *para* $x \neq y$ : |
| --- |
| $p_{\neq}(s) = \{x \mapsto \{x \mapsto s(x) - \mathit{unico}(s(y)),$ <br> $y \mapsto s(y) - \mathit{unico}(s(x))\}$ |

donde: $\begin{cases} \mathit{unico}(\{d\}) = \{d\} \\ \mathit{unico}(C) = \emptyset \quad \text{(cuando } C \text{ contiene más de un valor)} \end{cases}$

debe propagarse solamente cuando $x$ o $y$ se asigna

# Eventos

*Usuales* :

| | |
|---|---|
| $fijo(x)$ | $x$ resulta asignado |
| $min(x)$ | el mínimo de $x$ cambia |
| $max(x)$ | el máximo de $x$ cambia |
| $cualquier(x)$ | el dominio de $x$ cambia |

- las condiciones de los eventos no son excluyentes
  - si $fijo(x)$ ocurre: $min(x)$ o $max(x)$ ocurre, $cualquier(x)$ ocurre
  - si $min(x)$ o $max(x)$ ocurre: $cualquier(x)$ ocurre

# El conjunto de eventos

$$eventos(s, s') = \begin{cases} \{cualquier(x) \mid s'(x) \subset s(x)\} \cup \\ \{min(x) \mid min(s'(x)) > min(s(x))\} \cup \\ \{max(x) \mid max(s'(x)) < max(s(x))\} \cup \\ \{fijo(x) \mid |s'(x))| >= 1 \wedge |s(x)| > 1\} \end{cases}$$

- donde $s' \leq s$

# eventos: ejemplo

- Sea los stores:

$$s = \{x_1 \mapsto \{1,2,3\}, \quad \{x_2 \mapsto \{3,4,5,6\}$$
$$x_3 \mapsto \{0,1\}, \quad \{x_4 \mapsto \{7,8,10\}\ \}$$

$$s' = \{x_1 \mapsto \{1,2,3\}, \quad \{x_2 \mapsto \{3,4,5,6\}$$
$$x_3 \mapsto \{0,1\}, \quad \{x_4 \mapsto \{7,8,10\}\ \}$$

- entonces los eventos son:

$$\begin{aligned} eventos(s,s') = \quad & \{max(x_1), cualquier(x_1), \\ & cualquier(x_2), \\ & fijo(x_3), min(x_3), cualquier(x_3)\} \end{aligned}$$

# Propiedades de los eventos

- Los eventos son monotónicos:
  si $s_2 \leq s_1 \wedge s_1 \leq s$, entonces
  $eventos(s, s_2) = eventos(s, s_1) \cup eventos(s_1, s_2)$
- Eventos que ocurren al cambiar de $s$ a $s_2$:
  - los que ocurren al ir de $s$ a $s_1$
  - más los que ocurren al ir de $s_1$ a $s_2$

# Conjunto de eventos de propagador

Conjunto de eventos del propagador $p$: $ce(p)$

- Propiedad 1:
  - para todo store $s'$ y $s$ con $s' \leq s$ y $s'(x) = s(x)$ para toda variable que no está en $p$ (o sea, $x \in V - var(p)$),

$$\text{si } p(s) = s \wedge p(s') \neq s' \text{ entonces } ce(p) \cap eventos(s, s') \neq \emptyset$$

  - es decir, si el store $s$ es punto fijo y cambia a un no-puntofijo $s'$ entonces algunos eventos de $s$ a $s'$ deben estar incluidos en $ce(p)$

# Conjunto de eventos de propagador (2)

Conjunto de eventos del propagador $p$: $ce(p)$

- Propiedad 2:
  - para todo store $s$ con $p(p(s)) \neq p(s)$,

    $$ce(p) \cap eventos(s, p(s)) \neq \emptyset$$

  - es decir, si el propagador no calcula el punto fijo de $s$, entonces eventos de $s$ a $p(s)$ deben estar incluidos en $ce(p)$
  - obviamente, nunca se da en propagadores idempotentes

# El propagador de $\leq$

| *propagador* $p_{\leq}$ *para* $x \leq y$ : |
| --- |
| $p_{\leq}(s) = \{x \mapsto \{d \in s(x) \mid d \leq max(s(y))\}$ <br> $y \mapsto \{d \in s(y) \mid d \geq min(s(x))\}\}$ |

bueno: $ce(p_{\leq}) = \{max(y), min(x)\}$
posible: $ce(p_{\leq}) = \{cualquier(y), cualquier(x)\}$

# El propagador de $\neq$

| *propagador* $p_{\neq}$ *para* $x \neq y$ : |
| --- |
| $p_{\neq}(s) = \{x \mapsto \{x \mapsto s(x) - unico(s(y)),$ <br> $y \mapsto s(y) - unico(s(x))\}$ |

donde: $\begin{cases} unico(\{d\}) = \{d\} \\ unico(C) = \emptyset \quad \text{(cuando } C \text{ contiene más de un valor)} \end{cases}$

bueno: $ce(p_{\neq}) = \{fijo(y), fijo(x)\}$
posible: $ce(p_{\neq}) = \{cualquier(y), cualquier(x)\}$

# Utilizar eventos de propagador

basar decisión de re-propagación en conjuntos de eventos en lugar de modificación de variables

$$AP = \{q \in P \mid eventos(s, s') \cap ce(q) \neq \emptyset\}$$

# Descanso: regreso a modelamiento

**El problema de alinear personas para una foto**

1. Berta quiere estar al lado de Gastón y María
2. Cristina quiere estar al lado de Berta y Gastón
3. Felipe quiere estar al lado de María y Daniel
4. Pablo quiere estar al lado de Felipe y Daniel

El problema es sobre-restringido: no tiene solución exacta-

problema modificado:
encontrar solución que maximiza el número de preferencias satisfechas.
Se modelan mediante reificación de restricciones

# La especificación

- Variables:
  $pos_i$, la posición de la persona $i$ en la línea
  $sat$, el valor del número de preferencias satisfechas
- Restricciones:
  $|pos(pa) - pos(pb)| = 1$
  - La posición de la persona pa debe estar contigua a la de la persona pb
  - se supone que $pa$ al lado de $pb$ está en las preferencias.
- Las restricciones son reificadas:
  $(|pos(pa) - pos(pb)| = 1) \Leftrightarrow b_j$
  - $b_j$ es una variable booleana
  - si la restricción se cumple, $b_j = 1$ (y viceversa).

# La especificación: optimización

- se debe maximizar el número de preferencias satisfechas:

$$maximizar(\sum_{j} b_j)$$

- Expresar las preferencias:
  - $prefs = [p_1, p_2, ..., p_{2n}]$
  - para todo $i = 0, 2, ..., n-1$, la persona $prefs[2 \times i]$ prefiere estar al lado de la persona $prefs[2 \times i + 1]$
  - Ejemplo,

[Beatriz, Cristina,Daniel,Felipe,Gastón,María,Pablo]

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $prefs =$ | $[0,4,$ | $0,5,$ | $1,0,$ | $1,4,$ | $3,5,$ | $3,2,$ | $6,3,$ | $6,2]$ |

# El modelo

- Variables:

$$pos = new\ VarArray\ <IntVar>(this, s.n\_names(), IntVar.class, 0, spec.n\_names() - 1);$$
$$sat = new\ IntVar(this, new\ IntSet(0, spec.n\_prefs()));$$

- Restricciones:

$$int\ coefs[] = \{1, -1\};$$
$$VarArray\ <IntVar>\ args1 = new\ VarArray\ <IntVar>(pos.get(pb), pos.get(pa));$$
$$VarArray\ <IntVar>\ args2 = new\ VarArray\ <IntVar>(pos.get(pa), pos.get(pb));$$
$$linear(this, coefs, args1, IRT\_EQ, 1, b0, ICL\_DEF);$$
$$linear(this, coefs, args2, IRT\_EQ, 1, b1, ICL\_DEF);$$
// cump.get(i) es verdadero si $|pos(pa) - pos(pb)| = 1$
$$rel(this, b0, BOT\_XOR, b1, cump.get(i));$$

# El modelo (2)

- Otras restricciones:

```
distinct (this , pos, ICL_DEF);
rel (this , pos.get(0), IRT_LE, pos.get(1), ICL_DEF);
```

- maximización

```
// Suma de cumplimientos
{
  Expr sum = new Expr();
  for (int i = cump.size(); i - - > 0; ){
    sum.plus(cump.get(i));
   }
  post (this , sum, IRT_EQ, sat);
}
```