Curso de Doctorado:

Programación Internet con Lenguajes Declarativos Multiparadigma.

PARTE I: Fundamentos

Pascual Julián Iranzo

Pascual.Julian@uclm.es

Universidad de Castilla – La Mancha. Departamento de Informática.

Indice

1.- Introducción.

2.- Sistemas ecuacionales.

3.- Sistemas de reescritura de términos.

$\Rightarrow$ Narrowing, estrategias de narrowing y residuación.

Narrowing, estrategias de narrowing y residuación

Motivación y Objetivos

- Estudiar los principales mecanismos operacionales de los Lenguajes Lógico Funcionales:
  - **Narrowing** [= (Instanciación de variables / unificación) + Reescritura];
  - **Residuación** [= espera hasta que las llamadas a función estén suficientemente instanciadas + Reescritura].

Narrowing, estrategias de narrowing y residuación.

Narrowing

- Los lenguajes funcionales persiguen el cómputo de valores de expresiones básicas.
- Hemos visto como la reescritura sirve (en parte) para fundamentar los lenguajes funcionales.
- También hemos visto como la reescritura puede resolver el *(groung) word problem*.

Narrowing

- Sin embargo, los lenguajes funcionales no están bien equipados para tratar con expresiones que contienen variables o resolver el problema de la satisfacibilidad de ecuaciones.
- **Ejemplo**: Dado el TRS

$$R_1 : 0 \leq N \rightarrow true \qquad R_3 : s(M) \leq s(N) \rightarrow M \leq N$$

$$R_2 : s(M) \leq 0 \rightarrow false \qquad R_4 : dec(s(X)) \rightarrow X$$

el término $s(X) \leq Y$ es irreducible.

Narrowing

- **Ejemplo**: (continuación) El término $s(X) \leq Y$ puede reducirse a $true$ si sustituimos Y por $s(N1)$ y X por 0:

$$s(0) \leq s(N1) \quad \rightarrow_{R_3} \quad 0 \leq N1 \quad \rightarrow_{R_1} \quad true.$$

- El narrowing puede verse como un método que permite encontrar la sustitución “justa”, que aplicada a una expresión, la hace reducible.

Narrowing

- **Paso de narrowing**: Un término t se reduce por *narrowing* a s, escrito $t \overset{[p,R,\sigma]}{\rightsquigarrow} s$, si
  - existe una posición $p \in \mathcal{FP}os(t)$,
  - una (variante de una) regla $R : l \to r$
  - y una substitución σ tal que:

1. σ es un unificador de $t|_p$ y l), y
2. $s \equiv \sigma(t[r]_p)$.

Narrowing

- También escribimos $t \overset{\sigma}{\leadsto} s$ cuando no interesen los de talles de la posición, etc. con la que damos el paso.
- Cuando σ es el unificador más general (i.e., $\sigma = mgu(\{t|_p = l\})$) hablamos de que el paso de narrowing es más general.
- Que σ no sea un unificador más general es un requisito indispensable en la definición de algunas estrategias de *narrowing* (e.g., *needed narrowing*).

Narrowing

- **Derivación de Narrowing** de t a s: secuencia de pasos de narrowing

$$t \equiv t_0 \overset{[p_1,R_1,\sigma_1]}{\rightsquigarrow} t_1 \overset{[p_2,R_2,\sigma_2]}{\rightsquigarrow} \ldots \overset{[p_n,R_n,\sigma_n]}{\rightsquigarrow} t_n \equiv s$$

- **Notación**: $t \overset{\sigma}{\rightsquigarrow}{}^* s$, donde $\sigma = \sigma_n \circ \ldots \circ \sigma_2 \circ \sigma_1$.

- **Ejemplo**: Dado el TRS de la página 5.

$$\underline{s(X) \leq Y} \overset{[R_3,\{X/M1,Y/s(N1)\}]}{\rightsquigarrow} \underline{M1 \leq N1} \overset{[R_1,\{M1/0,N1/N2\}]}{\rightsquigarrow} true$$

Narrowing

- Los programas lógico funcionales permiten tanto el cómputo de un resultado como la obtención de una respuesta.
- El par $\langle s, \sigma \rangle$ es la salida de la derivación. El término s es el resultado y σ la respuesta (parcial).
- **Ejemplo**: Para la anterior derivación, la salida es: $\langle true, \{X/0, Y/s(N2), M1/0, N1/N2\} \rangle$.

Narrowing

- Hablaremos de derivación de éxito para referirnos a las derivaciones $t \overset{\sigma}{\leadsto}{}^* s$ cuyo resultado s es un valor.
- Un valor es una forma irreducible (en particular, un término constructor —no necesariamente básico—).
- Para una derivación de éxito, $t \overset{\sigma}{\leadsto}{}^* s$, $\sigma_{\restriction \mathcal{V}ar(t)}$, es una respuesta computada.

Narrowing

- **Ejemplo**: Para el ejemplo anterior, $\{X/0, Y/s(N2)\}$ es una respuesta computada.
- Hablaremos de derivación de fallo cuando el resultado de la derivación es una forma irreducible que no es un valor.
- **Ejemplo**:

$$s(s(X)) \leq s(dec(0)) \rightsquigarrow s(X) \leq dec(0) \not\rightsquigarrow$$

Narrowing

- Las derivaciones de *narrowing* para un término pueden representarse mediante un árbol de búsqueda (posiblemente infinito) ramificado finitamente.
- Un árbol de narrowing para t en $\mathcal{R}$ (denotado $\tau(t, \mathcal{R})$) es un conjunto de nodos que cumplen:

1. t es el nodo raíz de τ;
2. Si $t \in \tau$, para todo paso de *narrowing* $t \leadsto s$ $s \in \tau$.

Narrowing

- **Ejercicio**: Representar el árbol de *narrowing* para el término $t \equiv s(X) \leq Y$ y el TRS de la página 5.
- Cada una de las ramas del árbol de *narrowing* τ representan una de las posibles derivaciones de *narrowing* a partir del término situado en la raíz.
- Las *hojas* de τ representan el resultado de la computación.

Narrowing: el problema de la $\mathcal{E}$-unificación

- El *narrowing* se introdujo como un medio para resolver *ecuaciones*
- Las ecuaciones pueden manipularse como términos si ampliamos la signatura con los símbolos $\{\approx, true\}$ y suponemos que todo TRS (convergente) contiene la regla: $x \approx x \rightarrow true$
- Esta regla no destruye la confluencia y terminación del TRS ampliado.

Narrowing: el problema de la $\mathcal{E}$-unificación

- la regla $x \approx x \rightarrow true$ puede entenderse como un medio de integrar la unificación sintáctica en el procedimiento de *narrowing*.
- Dado un TRS $\mathcal{R}$ y la teoría ecuacional $\mathcal{E} = \{l = r \mid (l \rightarrow r) \in \mathcal{R}\}$, el *narrowing* es:

1.- **Correcto**: Si $(s \approx t) \overset{\sigma}{\leadsto}{}^{*} true$ es una derivación de *narrowing* entonces $\sigma(s) \approx_{\mathcal{E}} \sigma(t)$ (i.e., σ es un $\mathcal{E}$-unificador de s y t).

Narrowing: el problema de la $\mathcal{E}$-unificación

- Dado un TRS $\mathcal{R}$ y la teoría ecuacional $\mathcal{E} = \{l = r \mid (l \to r) \in \mathcal{R}\}$, el *narrowing* es:

  2.- **Completo**: Si $\sigma(s) \approx_{\mathcal{E}} \sigma(t)$ entonces existe una derivación de *narrowing* $(s \approx t) \overset{\theta}{\leadsto}{}^{*} true$ tal que $\theta \leq_{\mathcal{E}} \sigma\ [\mathcal{V}ar(s) \cup \mathcal{V}ar(t)]$.

- **Observación**: Bajo la condiciones para la corrección y completitud del *narrowing*, σ es una solución de una ecuación si y sólo si $(s \approx t) \overset{\sigma}{\leadsto}{}^{*} true$.

Narrowing: el problema de la $\mathcal{E}$-unificación

- En general el *narrowing* es correcto, sin embargo, para lograr la completitud deben imponerse ciertas restricciones, tanto a los TRS's como al tipo de resultados y respuestas obtenidos.
- En la prueba de la corrección es clave la observación de que: $t \overset{\sigma}{\leadsto} s \quad \Rightarrow \sigma(t) \rightarrow s$.
- El *narrowing* es completo para TRSs convergentes.

Narrowing: el problema de la $\mathcal{E}$-unificación

- En la prueba de la completitud se emplea el siguiente Lema de *lifting*:
  - s y t, términos
  - θ una sustitución normalizada y $t = \theta(s)$.
  - $\mathcal{V} \subset \mathcal{X}$, tal que $\mathcal{V}ar(s) \cup \mathcal{D}om(\theta) \subseteq \mathcal{V}$

  Si $t \rightarrow^* t'$ entonces, existe un s' y una θ', tal que
  - $s \overset{\sigma}{\rightsquigarrow}^* s'$,
  - $t' = \theta'(s')$, $\theta' \circ \sigma = \theta$ y θ' está normalizada.

Narrowing: el problema de la $\mathcal{E}$-unificación

- **Observación**: θ es una sustitución normalizada sii $\forall x \in \mathcal{D}om(\theta)$, $\theta(x)$ está en forma normal.
- El *narrowing* es completo para TRSs confluentes y sustituciones normalizadas.
- También, para sustituciones normalizadas, podemos eliminar el subíndice "$\mathcal{E}$" del orden "$\leq_{\mathcal{E}}$" y emplear el preorden de máxima generalidad "$\leq$" en la definición de completitud.

Estrategias de Narrowing

- En general, el procedimiento de *narrowing* es indeterminista, debido a la existencia de dos grados de libertad:
  - la elección del subtérmino a reducir y
  - la elección de la regla.
- Esto conduce a un espacio de búsqueda demasiado amplio (que debe ser recorrido en toda su amplitud para mantener la completitud del cálculo).

Estrategias de Narrowing

- Una estrategia de *narrowing* es una restricción del espacio de búsqueda.
- Una estrategia de narrowing es una aplicación φ que, para un término t, computa el conjunto de ternas $\langle p, R, \sigma \rangle$, donde
  - $p \in \mathcal{FP}os(t)$,
  - $R \equiv (l \rightarrow r)$ es la regla del TRS utilizada para dar el paso de *narrowing* y
  - σ una substitución unificadora de $t|_p$ y l

Estrategias de Narrowing

- $t \overset{[p,R,\sigma]}{\leadsto_{\varphi}} \sigma(t[r]_p)$ es un paso de *narrowing* que respeta la estrategia φ, si $\langle p, R, \sigma \rangle \in \varphi(t)$.
- Si el conjunto $\varphi(t)$ contiene un solo elemento, decimos que el paso de *narrowing* es determinista.
- Una derivación respeta la estrategia φ, y escribimos $t \overset{\sigma}{\leadsto_{\varphi}}{}^{*} s$, si cada uno de sus pasos es un paso de *narrowing* que respeta la estrategia φ.

Estrategias de Narrowing

- Se han diseñado muchas estrategias para reducir el tamaño del espacio de búsqueda, eliminando algunas derivaciones inútiles.
- Una propiedad importante de toda estrategia es que siga manteniendo la completitud del cálculo.
- Una clasificación de las diferentes estrategias y las condiciones para que sea correcta y completa, puede encontrarse en [Hanus 1994].

Estrategias de evaluación perezosa

- Nos centraremos en las estrategias de evaluación perezosa. Principalmente:
  - *narrowing perezoso* (*lazy narrowing* — LN) y
  - *narrowing necesario* (*needed narrowing* — NN).
- En este contexto,
  - programa = subclase de los TRSs CB y lineales por la izquierda y
  - la igualdad es una forma más débil de identidad (igualdad estricta).

Expresividad e igualdad estricta

- Antes de seguir adelante conviene precisar ciertos aspectos sintácticos y el concepto de igualdad estricta.
- Con el fin de aumentar la expresividad del lenguaje, extendemos la signatura $\mathcal{F}$ con un conjunto de símbolos de función primitivos: $\mathcal{P} = \{\approx, \wedge, \Rightarrow\}$.
- Esto permite manipular expresiones complejas (e.g., ecuaciones) como términos.

Expresividad e igualdad estricta

- La semántica de estos símbolos primitivos viene determinada por el siguiente conjunto de reglas predefinidas, que denotamos por StrEq:

$$
\begin{array}{rcll}
c \approx c & \rightarrow & true & \% \; c \in \mathcal{C}, ar(c) = 0 \\
c(\overline{x_n}) \approx c(\overline{y_n}) & \rightarrow & (x_1 \approx y_1) \wedge \ldots \wedge (x_n \approx y_n) & \% \; c \in \mathcal{C}, ar(c) = n \\
true \wedge x & \rightarrow & x & \\
false \wedge x & \rightarrow & false & \\
(true \Rightarrow x) & \rightarrow & x &
\end{array}
$$

Expresividad e igualdad estricta

- **Observaciones**:
  - El operador $\wedge$ es el operador booleno de conjunción.
  - El operador $\Rightarrow$ permite construir expresiones condicionales guardadas ($boolean \Rightarrow t$). No tiene correspondencia exacta con el condicional lógico.
  - Las dos primeras reglas definen la validez de una ecuación como la igualdad estricta entre términos.

Expresividad e igualdad estricta

- La igualdad estricta define dos términos s y t como idénticos sii tienen como forma normal el mismo término constructor (básico).
  - **Proposición**: $s \approx t$ sii $s \approx t$ se reduce a $true$.
- Es habitual definir la igualdad estricta en los lenguajes (funcionales) que permiten manipular cómputos que no terminan.

Expresividad e igualdad estricta

- La igualdad estricta no es una verdadera relación de identidad, ya que no posee la propiedad reflexiva.
  - **Ejemplo**: Para $\mathcal{R} = \{f(X) \rightarrow f(X)\}$, la evaluación de $f(0) \approx f(0)$ no termina.
- En lo que sigue supondremos que todo programa $\mathcal{R}$ se extiende con el conjunto de reglas StrEq. Las reglas StrEq son ortogonales.

Narrowing Perezoso

- El *narrowing* perezoso reduce las expresiones comenzando por las posiciones más externas (“outermost”) sobre las que se puede dar un paso de *narrowing*.
- Los pasos de *narrowing* sobre posiciones más internas solamente se realizan si son demandados y contribuyen a un paso de *narrowing* posterior sobre una posición más externa.

Narrowing Perezoso

- LN se fundamenta sobre un mecanismo de unificación lineal.
- El algoritmo de unificación lineal difiere con respecto al algoritmo de unificación sintáctica estándar en que:
  - una colisión entre un símbolo constructor y uno de operación, no se ve como un fracaso, sino como una demanda de una mayor evaluación de f.

Narrowing Perezoso

- LN se fundamenta sobre un mecanismo de unificación lineal.
- El algoritmo de unificación lineal difiere con respecto al algoritmo de unificación sintáctica estándar en que:
  - Debido a la linealidad de los patrones que aparecen en las lhs's, no se precisa de ningún mecanismo de "occur-check".

Narrowing Perezoso

- Un problema de unificación lineal es un par de términos: $\langle f(\overline{d_n}), f(\overline{t_n})\rangle$; donde $f(\overline{d_n})$ es un patrón lineal que no comparte variables con $f(\overline{t_n})$.

- Una configuración LU es un par (U, σ), donde U es un conjunto $\{d_1 \downarrow_1 t_1, \ldots, d_n \downarrow_n t_n\}$, siendo $d_1, \ldots, d_n$ términos constructores lineales que no comparten variables y σ una substitución.

Narrowing Perezoso: Relación de Unificación $\rightarrow_{\mathsf{LU}}$

- La mínima relación que satisface:

1. $(\{c(\overline{d_m}) \downarrow_u c(\overline{t_m})\} \cup U, \sigma) \rightarrow_{\mathsf{LU}} (\{\overline{d_m \downarrow_{u.m} t_m}\} \cup U, \sigma)$, donde $(c/m) \in \mathcal{C}, m \geq 0$.
2. $(\{x \downarrow_u t\} \cup U, \sigma) \rightarrow_{\mathsf{LU}} (\{x/t\}(U), \{x/t\} \circ \sigma)$, donde $t \notin$
3. $(\{d \downarrow_u x\} \cup U, \sigma) \rightarrow_{\mathsf{LU}} (\{x/d\}(U), \{x/d\} \circ \sigma)$.
4. $(\{c(\overline{d_m}) \downarrow_u c'(\overline{t_p})\} \cup U, \sigma) \rightarrow_{\mathsf{LU}} (\{\texttt{fail}\}, \sigma)$, donde $(c/m), (c'/p) \in \mathcal{C}$, $c \neq c'$, y $m, p \geq 0$.

Narrowing Perezoso: Relación de Unificación $\rightarrow_{\mathsf{LU}}$

- Dada una configuración (U, σ) irreducible, una posición u es demandada, si $(c(\overline{d_m}) \downarrow_u g(\overline{t_p})) \in U$.
- Sea $\Gamma \equiv \langle f(\overline{d_n}), f(\overline{t_n})\rangle$ un problema de unificación lineal y $(U_0, \sigma_0) \equiv (\{\overline{d_n \downarrow_n t_n}\}, id) \rightarrow^*_{\mathsf{LU}} (U, \sigma) \not\rightarrow_{\mathsf{LU}}$.

$$\mathsf{LU}(\Gamma) = \begin{cases} (\textsc{Succ}, \sigma) & \texttt{si } U = \emptyset \\ (\textsc{Fail}, \emptyset) & \texttt{si } U = \{\texttt{fail}\} \\ (\textsc{Demand}, P) & \texttt{en otro caso,} \end{cases}$$

$P =$ conjunto de posiciones demandadas.

Narrowing Perezoso

- **Estrategia de Narrowing Perezoso λ_{lazy}:**

$$\lambda_{lazy}(t) = \bigcup_{k=1}^{m} \lambda\_(t, \Lambda, R_k)$$

$$\lambda\_(t, p, R_k) = \texttt{ si } \mathcal{R}oot(l_k) \equiv \mathcal{R}oot(t|_p) \texttt{ entonces}$$

$$\texttt{en caso de que } \texttt{LU}(\langle l_k, t|_p \rangle) =$$

$$\begin{cases} (\texttt{Succ}, \sigma): & \{\langle p, R_k, \sigma \rangle\} \\ (\texttt{Fail}, \emptyset): & \emptyset \\ (\texttt{Demand}, P): & \bigcup_{q \in P} \bigcup_{k=1}^{m} \lambda\_(t, p.q, R_k \end{cases}$$

$$\texttt{sino } \emptyset$$

Narrowing Perezoso

- Informalmente, la estrategia $\lambda_{lazy}(t)$ consiste en:

1. Seleccionar una posición más externa p de redex para t.
2. Si el proceso de unificación lineal de $t|_p$ con una lhs de una regla tiene éxito, entonces dar el paso de narrowing.
3. Si produce un conjunto de posiciones demandadas P, entonces hallar $\lambda_{lazy}(t|_u)$ para todo $u \in P$.
4. en otro caso, se produce un fallo.

Narrowing Perezoso

- **Ejercicio**: Dado el TRS

$$\begin{aligned} 0 \leq N &\rightarrow true \\ s(M) \leq 0 &\rightarrow false \\ s(M) \leq s(N) &\rightarrow M \leq N \end{aligned}$$

$$\begin{aligned} 0 + N &\rightarrow N \\ s(M) + N &\rightarrow s(M + N) \end{aligned}$$

Construir el árbol de narrowing perezoso para el término $X \leq X + X$:

  - Señalar las derivaciones de éxito.
  - Indicar las respuestas y valores computados.

Narrowing Perezoso: Propiedades

- **Proposition**: Sea $s \overset{\sigma}{\leadsto}_{LN}{}^{*} t$, una derivación. Entonces, para toda $x, z \in Dom(\sigma_{\restriction \mathcal{V}ar(s)})$, con $x \neq z$, se cumple que $\sigma(x)$ y $\sigma(z)$:
  - son términos constructores lineales, y
  - no comparten variables.
- **Teorema**: El *narrowing* perezoso es completo con respecto a la igualdad estricta y substituciones constructoras, para programas CB ortogonales.

Narrowing Perezoso: Ventajas

- El LN permite el cómputo con estructuras de datos infinitas.
- **Ejemplo**: Dado el programa

$$from(N) \rightarrow [N|from(s(N)];$$
$$first(0, L) \rightarrow []; \quad first(s(N), [E|L]) \rightarrow [E|first(N, L)]$$

y la expresión $first(X, from(Y)) \approx [0]$, el LN computa una respuesta (cúal?) mientras que una estrategia de *narrowing* voraz entra en un "bucle infinito".

Narrowing Perezoso e igualdad estricta

- La igualdad estricta puede tener comportamientos anómalos en combinación con la estrategia de narrowing perezoso.
- **Ejemplo**:
  - La ecuación $s(X) \approx s(Y)$ produce infinitas soluciones.
  - Para ecuación $s(X) \approx s(s(Y))$ LN no proporciona solución (hay una única derivación infinita).

Narrowing Perezoso e igualdad estricta

- Sin embargo, es evidente que la sustitución $\{X/s(Y)\}$ es una solución de la ecuación $s(X) \approx s(s(Y))$ (obtenida por un simple paso de unificación sintáctica).
- Estos problemas se solucionan mediante la extensión de la relación de narrowing con una nueva regla: regla de aceleración.

Narrowing Perezoso e igualdad estricta

- **Regla de aceleración**:

$$\frac{(s \approx t) \quad \wedge \quad s,t \in \mathcal{T}(\mathcal{C},\mathcal{X}) \quad \wedge \quad \sigma = mgu(s,t) \neq fail}{s \approx t \overset{[\sigma]}{\leadsto}_{\varphi} true}$$

- **Regla de aceleración**: Los problemas que hemos comentado, también aparecen cuando la igualdad estricta se combina con la estrategia de narrowing necesario (compruébese).

Narrowing Perezoso: Problemas

- El ejercicio de la página 39 revela que, dado un término, LN puede dar pasos sobre diferentes redexes demandados por diferentes reglas.
- Esto puede causar:
  - dificultades en una implementación secuencial que use *backtracking*.
  - computaciones innecesarias.

Narrowing Perezoso: Problemas

- Una implementación secuencial debe gestionar:
  - Los puntos de vuelta atrás asociados a la existencia de diversos redexes.
  - Los puntos de vuelta atrás asociados a la elección de diferentes reglas (habituales en programación lógica).
- Los programas uniformes se introdujeron para intentar eliminar la aparición de diversos redexes. (ver [Julián 2000 — Cap. 5] para una introducción).

Narrowing Perezoso: Problemas

- El siguiente ejemplo ilustra el problema de los cómputos innecesarios y las derivaciones redundantes.
- **Ejemplo**: Dado el TRS de la página 39 son posibles las siguientes derivaciones que computan la misma salida:
  - $\underline{X \leq X + X} \overset{[R_1,\{X/0\}]}{\leadsto_{LN}} true$
  - $X \leq \underline{X + X} \overset{[R_4,\{X/0\}]}{\leadsto_{LN}} \underline{0 \leq 0} \overset{[R_1,\{\}]}{\leadsto_{LN}} true$

Narrowing Perezoso: Problemas

- La segunda derivación reduce una posición de redex (la 2), que no es necesario reducir.
- El *narrowing* necesario se introdujo para resolver este problema.
- El narrowing necesario se basa en la selección de las posiciones necesarias más externas de un término, de modo que solamente se dan pasos inevitables para el cómputo de un resultado.

Narrowing Necesario: Arboles definicionales

- NN se define para la clase de los programas inductivamente secuenciales.
  - Son un subconjunto de los programas CB y ortogonales.
  - Coinciden con la clase de los TRSs CB y fuertemente secuenciales
- Su definición precisa se basa en el concepto de árbol definicional.

Narrowing Necesario: Arboles definicionales

- Dado un TRS $\mathcal{R}$, $\mathcal{P}$ es un árbol definicional parcial con patrón π sii:

1. $\mathcal{P} = rule(\pi, l \to r)$, donde $(l \to r) \in \mathcal{R}$ y π es una variante de l.
2. $\mathcal{P} = branch(\pi, o, \overline{\mathcal{P}_k})$, donde o es una posición de variable de π, para $k > 0$, $\overline{c_k}$ son constructores distintos y $\mathcal{P}_i$ es un árbol definicional parcial con patrón $\pi[c_i(\overline{x_n})]_o$, donde $\overline{x_n}$ son variables nuevas.

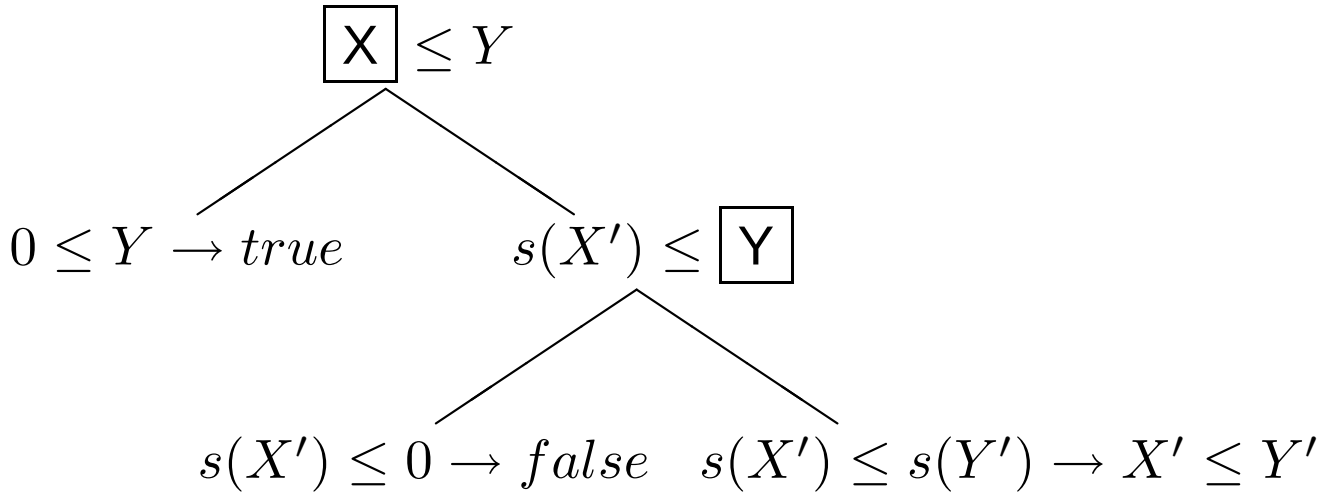
Narrowing Necesario: Arboles definicionales

- Si f es una función definida, $\mathcal{P}$ es un árbol definicional de f, sii es un árbol definicional parcial con $pattern(\mathcal{P}) = f(\overline{x_n})$, donde $\overline{x_n}$ son variables distintas.
- **Ejemplo**:árbol definicional para la función "$\leq$".

$$branch(X_1 \leq X_2, 1,$$
$$rule(0 \leq X_2, R_1),$$
$$branch(s(X_3) \leq X_2, 2, rule(s(X_3) \leq 0, R_2),$$
$$rule(s(X_3) \leq s(X_4), R_3)))$$

Narrowing Necesario: Arboles definicionales

- **Representación gráfica**: árbol definicional para la función "$\leq$".



Narrowing Necesario

- **Estrategia de Narrowing Necesario** ($\lambda(t, \mathcal{P})$): t un término; $\mathcal{P}$ un árbol definicional con $pattern(\mathcal{P}) = \pi$ tal que $\pi \leq t$.

  1. Si π es una hoja, entonces $\lambda(t, \mathcal{P}) = \{\langle \Lambda, \pi \rightarrow r, id \rangle\}$.

  2. Si π es una rama, considerar la posición inductiva o de π y un hijo $\pi_i = \pi[c_i(x_1, \ldots, x_n)]_o \in \mathcal{P}$. Entonces :

Narrowing Necesario

$$\lambda(t,\mathcal{P}) \ni \begin{cases} \langle p, R, \sigma \circ \tau \rangle & \texttt{si } t|_o = x \in \mathcal{X},\ \tau = \{x/c_i(x_1,\ldots,x_n)\}, \\ & \quad \texttt{y } (p, R, \sigma) \in \lambda(\tau(t), \mathcal{P}_i); \\ \langle p, R, \sigma \circ id \rangle & \texttt{si } t|_o = c_i(t_1,\ldots,t_n) \texttt{ y } (p, R, \sigma) \in \lambda(t, \mathcal{P}_i) \\ \langle o.p, R, \sigma \circ id \rangle & \texttt{si } t|_o = f(t_1,\ldots,t_n),\ f \in \mathcal{F} \\ & \quad \texttt{y } (p, R, \sigma) \in \lambda(t|_o, \mathcal{P}') \\ & \quad \text{donde } \mathcal{P}' \text{ es un árbol definicional para } \end{cases}$$

Narrowing Necesario

- Informalmente, el *narrowing* necesario:

1. aplica una regla si es posible (caso 1).
2. comprueba los subtérminos de las posiciones inductivas de una rama (caso 2):
   - si el subtérmino es una variable, queda instanciado con el constructor de un hijo (substitución adelantada);

y se evalúa aplicando recursivamente la definición de estrategia NN.

Narrowing Necesario

- Informalmente, el *narrowing* necesario:

1. aplica una regla si es posible (caso 1).
2. comprueba los subtérminos de las posiciones inductivas de una rama (caso 2):
   - si es un constructor, procede con el término y el subárbol hijo que corresponde al mismo constructor;

y se evalúa aplicando recursivamente la definición de estrategia NN.

Narrowing Necesario

- Informalmente, el *narrowing* necesario:

1. aplica una regla si es posible (caso 1).
2. comprueba los subtérminos de las posiciones inductivas de una rama (caso 2):
   - si es una función, procede con el término y el árbol que corresponde a esa función;

y se evalúa aplicando recursivamente la definición de estrategia NN.

Narrowing Necesario

- Esta estrategia difiere de otras estrategias perezosas en que cómputa respuestas que no son unificadores más generales.
- Suponemos que los árboles definicionales siempre contienen variables frescas cuando se emplean en el cómputo de un paso de NN.
- Esto supone que todas las substituciones computadas son idempotentes.

Narrowing Necesario

- Para cada $\langle p, l \to r, \sigma \rangle \in \lambda(t, \mathcal{P})$, el paso $t \overset{p,R,\sigma}{\leadsto}_{NN} \sigma(t[r]_p)$ es un paso de narrowing necesario.
- **Ejemplo**: Para el TRS que define "$\leq$" y "$+$" y el término $X \leq X + X$ existen los siguientes pasos de NN:
  - $X \leq X + X \overset{\{X/0\}}{\leadsto}_{NN} true$
  - $X \leq X + X \overset{\{X/s(M)\}}{\leadsto}_{NN} s(M) \leq s(M + s(M))$

Narrowing Necesario: Propiedades

- **Proposición**: Si $t \overset{\sigma}{\leadsto}_{NN}{}^{*} s$ es una derivación de NN, entonces para todo par de variables distintas $x, y \in \mathcal{D}om(\sigma_{\restriction \mathcal{V}ar(t)})$, $\sigma(x)$ y $\sigma(y)$ son términos constructores lineales que no comparten variables.
- **Teorema**: Para programas inductivamente secuenciales, el NN es correcto y completo con respecto a ecuaciones estrictas y substituciones constructoras como solución de dichas ecuaciones.

Narrowing Necesario: Propiedades

- **Minimalidad**: El NN no computa soluciones redundantes.

Si $e \overset{\sigma}{\leadsto}_{NN}{}^{*} true$ and $e \overset{\sigma'}{\leadsto}_{NN}{}^{*} true$ son dos derivaciones de NN distintas, entonces σ y σ' son independientes (i.e., existe al menos un $x \in \mathcal{V}ar(e)$ tal que $\sigma(x) \neq \sigma'(x)$).

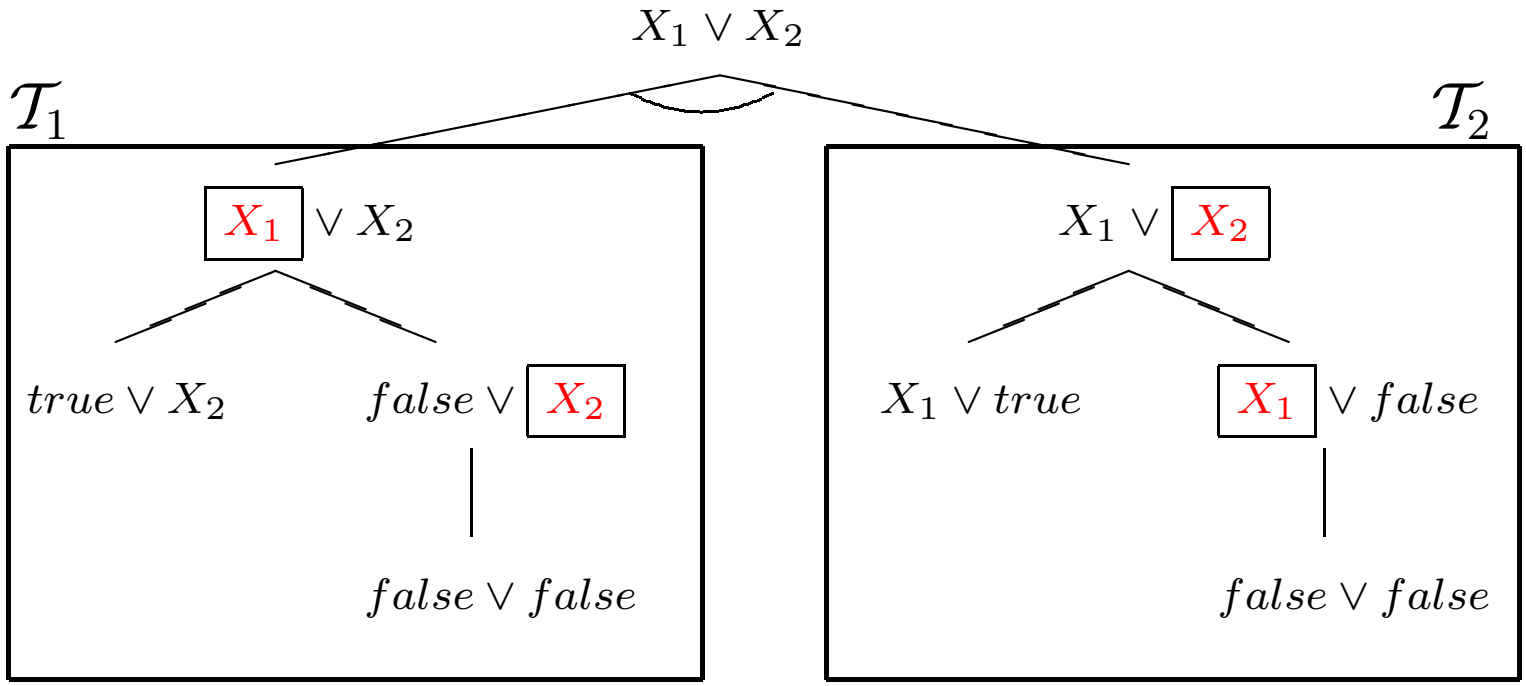
Narrowing Necesario: Extensiones

- El NN puede extenderse a programas CB y débilmente ortogonales mediante mediante la introducción de nodos or en los árboles definicionales.
- **Ejemplo**: Dado el TRS que define la operación *parallel–or*

$$X \vee true \rightarrow true$$
$$true \vee X \rightarrow true$$
$$false \vee false \rightarrow false$$

Narrowing Necesario: Extensiones

- **Ejemplo**: el árbol definicional asociado es,



Narrowing Necesario: Extensiones

- Los subárboles $\mathcal{T}_1$ y $\mathcal{T}_2$ son los componentes secuenciales del árbol definicional paralelo.
- El nodo–or señala la necesidad de evaluar, indeterministamente, un término con respecto a todas las componentes secuenciales.
- La generalización de NN a árboles definicionales paralelos se denomina narrowing necesario débil (*weakly needed narrowing*).

Narrowing Necesario: Extensiones

- El narrowing necesario débil ($\overline{\lambda}$) se define como:

$$\overline{\lambda}(t, \mathcal{T}) = \{\langle p, R, \sigma\rangle \in \lambda(t, \mathcal{T}') \mid \mathcal{T}' \text{ es un componente secuencial de } \mathcal{T}\}$$

- El narrowing necesario débil es correcto y completo para la igualdad estricta y sustituciones constructoras.
- El narrowing necesario débil no es óptimo.

Narrowing Necesario: Extensiones

- **Ejemplo**: Para el programa parallel–or y el término

$$(true \vee (true \vee true)) \vee (X \vee (false \vee false))$$

el NN débil computa seis veces la resp. $\{X/a\}$.

- El NN débil es indeterminista incluso para términos básicos: backtraking sobre términos básicos
- La estrategia de narrowing paralelo puede evitar estos problemas (ver [Antoy et al. 1997]).

Narrowing, estrategias de narrowing y residuación.

Residuación

- **Idea**:

  retrasar el cómputo de las llamadas a función hasta que están suficientemente instanciadas y listas para una reducción (determinista) por reescritura.

- Los mecanismos de búsqueda indeterminista quedan recluidos en los predicados.

- Las funciones preservan su naturaleza determinista.

Residuación

- **Ejemplo**: Dado el programa

$$0 + X \rightarrow X; \qquad nat(0) \rightarrow true;$$
$$s(X) + Y \rightarrow s(X + Y); \qquad nat(s(X)) \rightarrow (nat(X) \Rightarrow tru$$

  - La evaluación del término $Z + 0 \approx s(0)$ queda en suspenso.
  - La evaluación del término $Z + 0 \approx s(0)\ \&\ nat(Z)$ da la respuesta $\{Z/s(0)\}$

Residuación

- **Ejemplo** (Cont.):

$$
\begin{aligned}
& Z + 0 \approx s(0) \;\&\; \underline{nat(Z)} \\
\overset{\{Z/s(X1)\}}{\rightsquigarrow} \; & \underline{s(X1) + 0} \approx s(0) \;\&\; (nat(X1) \Rightarrow true) \\
\overset{\{Y2/0\}}{\rightsquigarrow} \; & \underline{s(X1 + 0) \approx s(0)} \;\&\; (nat(X1) \Rightarrow true) \\
\overset{\{X3/X+0,Y3/0\}}{\rightsquigarrow} \; & X1 + 0 \approx 0 \;\&\; (\underline{nat(X)} \Rightarrow true) \\
\overset{\{X1/0\}}{\rightsquigarrow} \; & \underline{0 + 0} \approx 0 \;\&\; (true \Rightarrow true) \\
\overset{\{X4/0\}}{\rightsquigarrow} \; & 0 \approx 0 \;\&\; (true \Rightarrow true) \rightsquigarrow^{*} true
\end{aligned}
$$

Residuación: Problemas

- El principio de residuación es incompleto.
- **Ejemplo**: La evaluación del término $Z + 0 \approx s(0)$ queda en suspenso, pero sabemos que esta exuación tiene como respuesta $\{Z/s(0)\}$.
- La residuación puede presentar un espacio de búsqueda infinito en situaciones en las que el narrowing computa uno finito.

Residuación y árboles definicionales

- El principio de residuación puede formalizarse mediante el uso de árboles definicionales.
- Extensión de los árboles definicionales:
  - ramas etiquetadas como rigidas (suspender expresiones con variables no suficientemente instanciadas).
  - ramas etiquetadas como flexibles (no suspender: aplicar narrowing).
  - inclusión de nodos and (conjunción concurrente).

Residuación y árboles definicionales

- La conjunción concurrente se define mediante las reglas:

$$R_1 : \ true \ \& \ X \rightarrow X; \qquad R_2 : \ X \ \& \ true \rightarrow X$$

- Se le asocia el árbol definicional:

$$\begin{aligned} and(\quad & branch(X_1 \& X_2, 1, rigid, rule(true \ \& \ X, R_1)), \\ & branch(X_1 \& X_2, 2, rigid, rule(X \ \& \ true, R_2))), \end{aligned}$$

Residuación y árboles definicionales

- Los nodos and provocan el siguiente orden de evaluación: Para una expresión $t_1 \& t_2$
  - primero, se intenta evaluar t_1.
  - Si t_1 suspende, se evalúa t_2.
- Esta aproximación permite unir los principios operacionales del narrowing y la residuación en un mismo marco (Ver los detalles en [Hanus 1997]).

Bibliografía

- Alpuente M., Falaschi M., Julián P. y Vidal G., 2003. Uniform Lazy Narrowing. *Journal of Logic and Computation*, 13(2):27.
- Antoy S., 1992. Definitional Trees. En *Proc. of the 3rd Int'l Conference on Algebraic and Logic Programming, ALP'92*, págs. 143–157. Springer LNCS 632.
- Antoy S., Echahed R. y Hanus M., 2000. A Needed Narrowing Strategy. En *Journal of the ACM*, volumen 47(4), págs. 776–822.

Bibliografía

- Antoy S., Echahed R. y Hanus M., 1997. Parallel Evaluation Strategies for Functional Logic Languages. En *Proc. of the 14th Int'l Conference on Logic Programming, ICLP'97*, págs. 138–152. MIT Press.
- Hanus M., 1997. A Unified Computation Model for Declarative Programming. En *Proc. of the 24th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, págs. 80–93. ACM.

Bibliografía

- Middeldorp A., Hamoen E., 1994. Completeness Results for Basic Narrowing. En *Journal of Applicable Algebra in Engineering, Communication and Computing*, volumen 5, págs. 213–253.