Sistemas de Reescritura

Cristóbal Domínguez Castro

Jorge García de la Nava Ruiz

Michel Pilíougine Rocha

Miguel Ángel Rico Blanco

18 de Junio de 2001

Introducción

Comenzamos con conceptos ya estudiados anteriormente. Entre estos conceptos tenemos los siguientes: Ecuaciones matemáticas -Haskell -Prolog -lambda-calculo -Maquina de Turing.

Ecuaciones Matemáticas

Para las ecuaciones matemáticas lo que nos interesa es ver como se aplica la igualdad para reducir expresiones complejas en otras mas simples para realizar un cómputo.

$\ln \sqrt{3 \cdot e^{16}}$

$=! \sqrt{x} = x^{1/2}$ ← Pasamos de una expresión a otra gracias a unas igualdades previamente demostradas. Este paso cumple la propiedad de **transparencia referencial** TODO VALE LO MISMO.

$\ln(3 \cdot e16)^{1/2}$

$=! \ln a^b = b \cdot \ln a$

$\frac{1}{2} \cdot \ln(3 \cdot e^{16})$

$=$

$\ldots\ldots$

↓ Se aplican las igualdades para que la expresión sea "más simple". Finalmente se llega a un valor irreducible.

La utilización de estas igualdades es muy ambigua, no sabemos que ecuación aplicar ni sobre que expresión, ni en que orden.... De modo que un humano puede hacerlo de manera intuitiva, pero un computador no tiene esta habilidad. Al ser las igualdades bidireccionales, podrían darse casos como el siguiente, provocando derivaciones infinitas.

$\ln \sqrt{3 \cdot e^{16}}$

$=! \sqrt{x} = x^{1/2}$

$\ln(3 \cdot e16)^{1/2}$

$=! \sqrt{x} = x^{1/2}$

$\ln \sqrt{3 \cdot e^{16}}$

Haskell

El caso de Haskell plantea, frente a las ecuaciones, una simplificación importante: la unidireccionalidad de sus reglas.

```
1. And 0 x = 0
2. And 1 x = x
3. Or 0 x = x
4. Or 1 x =1
```

→

La igualdad es en un sentido

```
and ( or (and 1 0) 1 ) (or 0 (and 1 1))
```

=! Por 1
`and (or 0 1) (or 0 (and 1 1))`

=! Por 3
.....

(Al final llegamos a un valor irreducible, es decir, ya no se pueden aplicar mas reglas)

Se presentan otros problemas como la elección de la subexpresión o el de elección de la regla a aplicar. En el ejemplo, si decidimos aplicar la regla 1 tenemos que decir a que subexpresión (tenemos dos posibilidades `and 1 0` y `and 1 1`. Por otro lado, podríamos haber aplicado en primer lugar la regla 3 y simplicar así el segundo argumento. Entonces, nos enfrentamos a nuevas preguntas: ¿Cambia el resultado final según el orden de aplicación las reglas? ¿ y según la subexpresión?

Prolog

Con Prolog tenemos dos variantes sobre los apartados anteriores: la unificación, que es una sustitución en ambos sentidos y el backtracking, que permite calcular distintos resultados para una misma expresión.

suma(0,X,X)
suma(s(y),X,s(Z)):- suma(Y,X,Z)

Veamos primero el caso de expresiones sin varibles (sólo términos básicos).

Suma(s(s(0)),s(s(0)),s(s(s(s(0)))))
=
Suma(s(0),s(s(0)),s(s(s(0))))
=
.....

Aquí también se aplican reglas buscando un término irreducible. En este caso ese término es único y se denomina cláusula vacía .

Si se prohíbe la unificación y el backtracking, tenemos un sistema de reescritura puro. Si se permiten, no obtenemos un sistema de reescritura puro, sino que habría que incluir elementos extras que simulen estas características.

Por ejemplo:

Suma(s(0),s(0),X)
`Unificación: X=s(Z1)`

Suma(0,s(0),Z1)
`Unificación: Z1=s(0)`

Para que esta derivación sea un sistema de reescritura habría que guardar el valor de las variables en la misma cadena. Para el backtracking habría que guardar la pila de la configuración.

λ-Cálculo

El λ-calculo es un sistema particular de reescritura y ya se ha estudiado en esta asignatura. Este documento generaliza muchos de los resultados del lambda-calculo y, en cierta manera, justificarlos en una teoría mas amplia y abstracta.

Máquinas de Turing

Estado	Símbolo	Acción	Siguiente Estado
1	0	L	1
1	1	0	1
1		R	2
2		R	3
3	0	R	3
3		PARAR	

Entonces, suponiendo la configuración Inicial dada, tenemos la siguiente derivación:

1.- (1, {.... 0111 ....})
2.- (1, {.... 011**0** ....})
3.- (1, {.... 01**1**0 ....})
4.- (1, {.... 0**1**00 ....})
5.- (1, {.... 0**1**00 ....})
6.- (1, {.... 0**0**00 ....})
7.- (1, {.... **0**000 ....})
8.- (1, {.... _0000 ....})
9.- (2, {.... **0**000 ....})
10.- (3, {.... **0**000 ....})
11.- (3, {.... 0**0**00 ....})
12.- (3, {.... 00**0**0 ....})
13.- (3, {.... 000**0** ....})
14.- (3, {.... 0000_ ....})
PARAR

Esto es extensible al caso determinista e indeterminista ya que son equivalentes. Como caso particular de un modelo de computación tenemos la ejecución de un programa en un computador real, que podría ser visto como un sistema de reescritura del contenido de la memoria.

Finalizando, un sistema de escritura, desde el punto de vista mas abstracto, puede ser visto como un grupo de relaciones (reglas) sobre un conjunto. Dependiendo de estas reglas y de este conjunto tendremos sistemas de reescritura particulares.

$$A = (A, \{\rightarrow, \rightarrow, \rightarrow, \rightarrow, ...\})$$

Definición ARS

Sea ***A*** un Sistema Abstracto de Reescritura definido de la forma siguiente ***A***=(A,($\rightarrow_{R1,}$ $\rightarrow_{R2}$,.... $\rightarrow_{Rn}$)), donde A es un conjunto sobre el que se dan la relaciones de reescritura o reducción $\rightarrow_{R1,}$ $\rightarrow_{R2}$,.... $\rightarrow_{Rn}$

NOTACIÓN
Supongamos definida sobre A la relación de igualdad = como nosotros la conocemos, en ambos sentidos.

$\rightarrow_R$	$\equiv$ R	(Notaremos la relación con o sin flecha)
$\rightarrow^*$	$\equiv \twoheadrightarrow$	(cero o + pasos) (Clausura Reflexiva Transitiva)
$\rightarrow^{-1}$	$\equiv \leftarrow$	(Inversa)
$\rightarrow^+$		(Cierre Transitivo)
$\rightarrow^=$		(Cierre Reflexivo)
$=_R$		según la relación R (relación de convertibilidad según R ***en este sentido***). Esta igualdad es generada a partir de la relación R.

Veamos que esta relacion $=_R$ es de ***equivalencia***

$a \twoheadrightarrow_R b \Rightarrow a =_R b$ (Definición de $=_R$ q también indica simetría en ese sentido) Cuando dos elementos están relacionados, como la relación es de reescritura, pues son iguales.
$a =_R b \Rightarrow b =_R a$ (Reflexividad en ambos sentidos, que realmente es igualdad)
$a =_R b \wedge b =_R c \Rightarrow a =_R c$

Como resultado podemos decir que nos queda un sistema conexo.

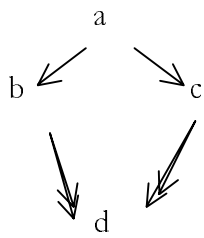
OTRAS DEFINICIONES
Forma normal:
â es Forma Normal $\Leftrightarrow \nexists$ b $\in$ A : â $\rightarrow$ b.
Notaremos las formas normales con ^:

Tener forma normal:
b $\in$ A tiene Forma Normal si $\exists$ â $\in$ A: b $\twoheadrightarrow$ â (y â es forma normal)

Relación De Propiedades

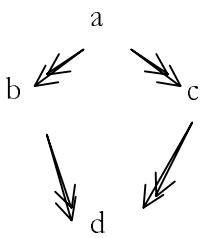
Confluencia Débil WCR(→)

$\forall$ a, b, c : a, b, c $\in$ A : $\exists$ d : d $\in$ A



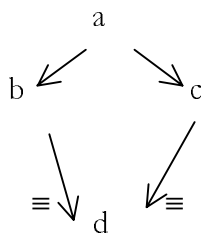
Confluencia CR(→)

$\forall$ a, b, c : a, b, c $\in$ A : $\exists$ d : d $\in$ A



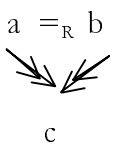
Subconmutativa WCR$^{\leq 1}$

$\forall$ a, b, c : a, b, c $\in$ A : $\exists$ d : d $\in$ A

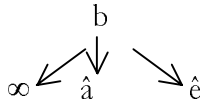


Confluencia por Equivalencia CEq($\rightarrow_R$)

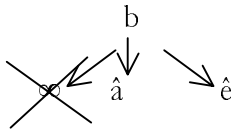
$\forall$ a, b : a, b $\in$ A : $\exists$ c : c $\in$ A



Débilmente Normalizante WN(→).
$\forall b : b \in A : \exists \hat{a} : \hat{a} \in A : b \rightarrow \hat{a}$



Fuertemente Normalizante SN(→)
Indica WN y Terminación



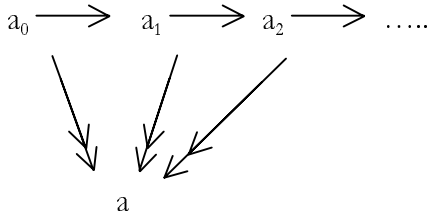
Únicamente Normalizante UN($\rightarrow_R$)
$\forall \hat{a}, \hat{e} : \hat{a}, \hat{e} \in A : \hat{a} =_R \hat{e} \Rightarrow \hat{a} = \hat{e}$

Incremental Inc(→)
$\exists f : f:A \rightarrow N : \forall a, b : a, b \in A : a \rightarrow b \Rightarrow f(a) < f(b)$

Notas.

- N es el conjunto de los naturales
- Esta propiedad refleja que un elemento al que se llega por una relación tendrá un numero asociado mayor que los anteriores, y por tanto las formas normales de una cadena de elementos relacionados tendrán un numero asignado mayor que cualquiera de sus antecedentes.

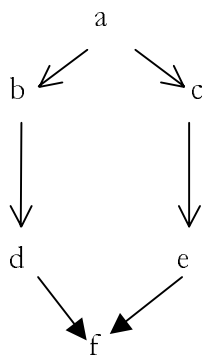
Inductividad Ind(→)
Sea una cadena infinita de la forma a → b → c → , entonces se da que
a↠ς, b↠ς, c↠ς, ……



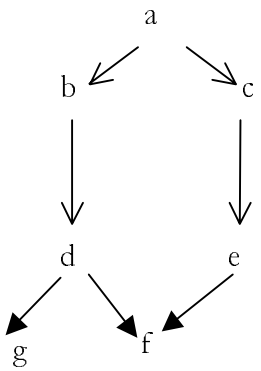
EJEMPLOS

Con los siguientes ejemplos intentemos aclarar algunas propiedades.

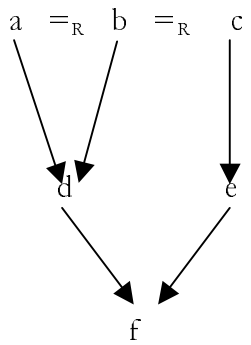
Veamos en primer lugar la diferencia entre las propiedades WCR y CR. El siguiente ejemplo cumple ambas propiedades:



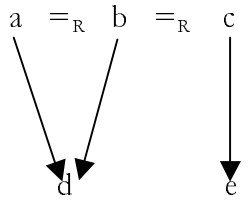
Sin embargo, al añadir otro elemento g, relacionado con d, la propiedad CR deja de cumplirse, ya que la propiedad CR indica que todo elemento alcanzable desde la raiz, debe de alcanzar a un único elemento, y en este caso el elemento d no lo cumple. Pero la propiedad WCR sigue cumpliéndose. Veámoslo:



Vamos a continuación a ver otro ejemplo, para aclarar la propiedad de la CEq (Confluencia por Equivalencia). Primero veremos una relación que sí cumple dicha propiedad:



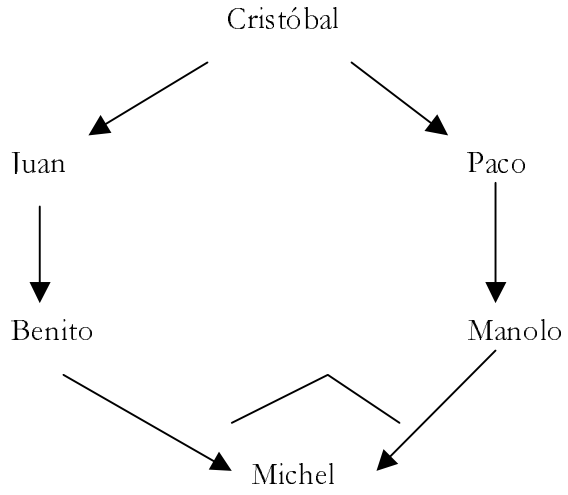
Veamos ahora, un ejemplo que no cumple la propiedad de Ceq. Bastaría con eliminar el elemento f. Ya no habría un elemento alcanzable por todos los demás. Veámoslo:



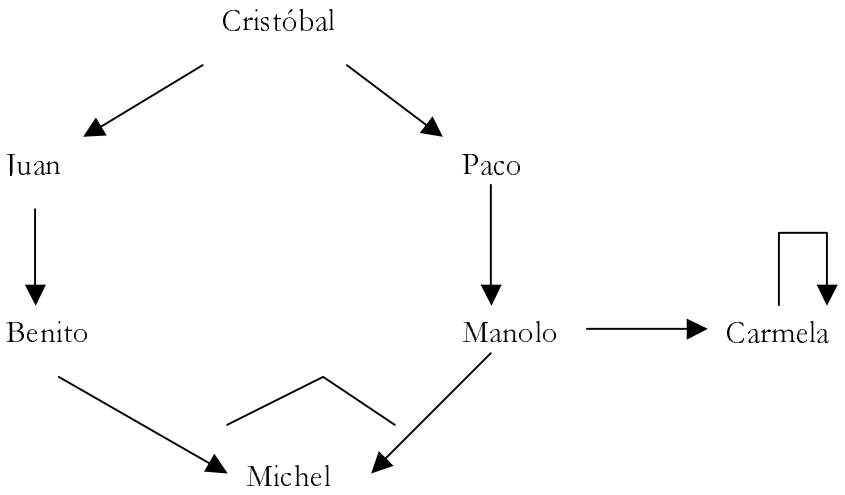
EJEMPLO DE CLASE

Para esclarecer las demás propiedades vamos a desarrollar a continuación el ejemplo visto en la exposición que se hizo en clase. Hemos de reconocer que es un ejemplo cómico, pero sin duda, no falto de la didáctica suficiente para hacer más amena la comprensión de estas propiedades. Expliquemos paso a paso su funcionamiento:

- El ejemplo lo situamos en el día anterior a un examen de la asignatura Lenguajes de Programación, que como sabemos todos no es nada fácil de aprobar.
- Es aquí cuando nuestro compañero Michel, se da cuenta, de que no tiene conocimientos suficientes para aprobar la asignatura del examen y le pide a su compañero Cristóbal el favor de que le pase un folio en el examen con la solución a las preguntas.
- Cristóbal, no sin antes recapacitar concienzudamente y sopesar su decisión, antepone su amistad con Míchel a la posibilidad de ser expulsado del examen, y decide que va a pasarle el folio con las soluciones.
- Llegado el día del examen, ambos compañeros no tienen la suerte de estar sentados juntos, porque la asignación de asientos se hace por orden alfabético.
- Sin embargo tienen la suerte de que conocen a todas las personas que existen entre ambos, y los convencen para que sirvan de paso del folio que llevará la información desde Cristóbal hasta Míchel.
- El sistema de reescritura, por tanto, lo podemos entender como que las personas que están en el examen cercanas a nuestros protagonistas, son los elementos que están relacionados entre sí, mediante la relación de PASAR_FOLIO. Volvemos a hacer incapié en que el ejemplo es un poco sui generis, pero estamos seguros de que ayudará al lector a comprender las propiedades que intentamos explicar de manera más amena. El sistema de reescritura queda de la siguiente manera:

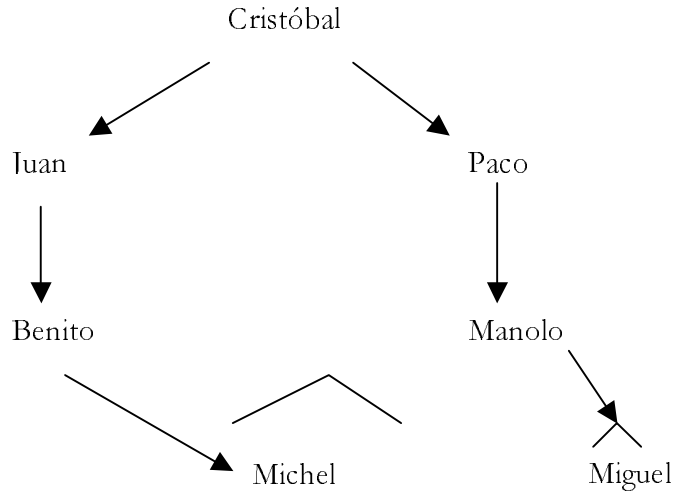


- Como se puede apreciar en la figura anterior Michel es la forma normal del sistema de reescritura.
- El folio que se pasa entre los alumnos debe de hacerse entre los alumnos relacionados por la regla → entendida como que el folio se reescribe. Por ejemplo de Cristóbal a Juan.
- Es fácil observar que este sistema cumple la propiedad de Confluencia (CR) y la Fuertemente Normalizante (SN), porque vaya por donde vaya el folio, llegará a Michel, y será forma normal. También cumple la propiedad Únicamente Normalizante (UN), porque existe una única forma normal que sera Míchel.
- Sin embargo, ocurre un imprevisto. Existe una persona que aún no ha aparecido y es Carmela, que es la novia de Manolo, y que también lleva el examen mal preparado.
- Entonces Manolo decide pasarle el folio a ella, y esta se lo queda:



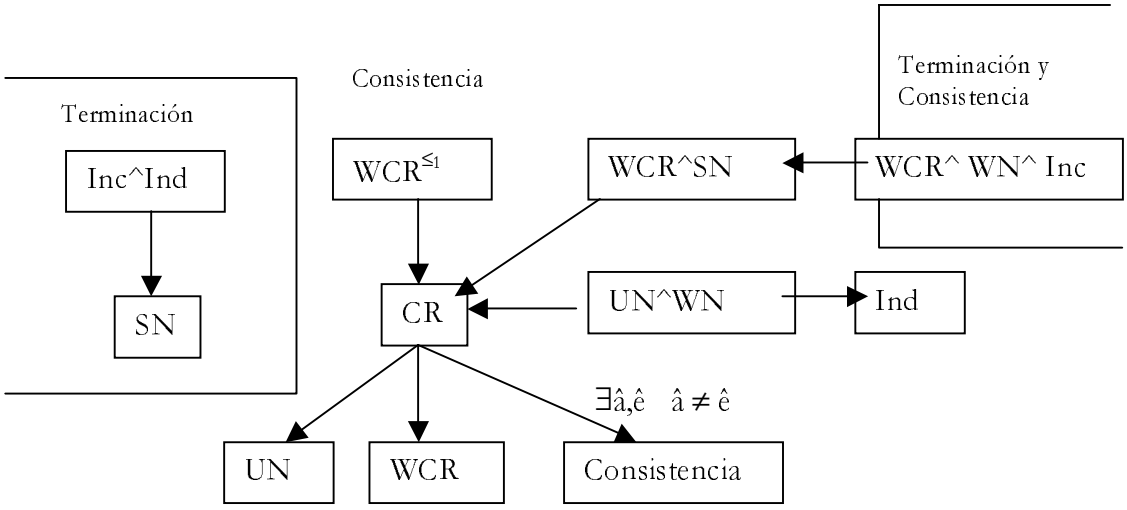
- El sistema resultante ya no es fuertemente normalizante (SN) y pasa a ser débilmente normalizante (WN) porque para Carmela no existe forma normal, ya que ella no va a devolver el folio que le pasó Manolo.

- Volvamos al sistema original.
- Y ahora supongamos que Manolo cree que Michel es otra persona, Miguel. Entonces vemos que existirán dos personas que serán formas normales, Michel y Miguel. El sistema queda como sigue.



- Entonces el sistema ya no cumple la propiedad UN (Unica forma normal). Tampoco cumple la propiedad WCR y por ende tampoco cumple la de CR.

Grafo Con La Relacion Entre Propiedades



Comentemos el cuadro:

- La propiedad de Consistencia cumple que: $\exists\ x,y \in A : x \neq_R y$
- Lo más importante para un sistema es que haya consistencia, para que haya clases de equivalencia, porque si no todos los elementos serian el mismo y no podríamos deducir nada.
- **WCR∧WN∧Inc ⇒ WCR ∧ SN** . En este paso hemos eliminado las cadenas infinitas.
- **UN∧WN ⇒ CR** . También se eliminan las cadenas infinitas. Al exigir una única forma normal, hacemos que termine siempre.
- **CR ⇒ Consistencia**. Si fijamos una única forma normal, y CR (confluencia) pues es obvio que tenemos consistencia porque así, los elementos son distintos entre sí.
- **La Confluencia es la base de la Consistencia.** Es fácil comprobar la veracidad de esta afirmación tanto desde un análisis de las propiedades, así como de un análisis de la ubicación de la propiedad de confluencia (CR) en el grafo. La propiedad CR está ubicada en el centro del grafo, o sea, es la propiedad "comodín" para que el grafo tenga sentido, y también es necesaria para la consistencia.

Definición de TRS

Sea $\mathcal{T}$ un sistema de reescritura de términos (TRS) que se define de la forma siguiente:

$\mathcal{T} = (\Sigma, (\mapsto_{R1}, \mapsto_{R2}, \cdots \mapsto_{Rn}))$

donde Σ es un alfabeto de términos formado por:

Σ_V = Alfabeto de variables (x, y, z…)
Σ_F = Alfabeto de funtores (F, G, H…) cada uno con una aridad dada.
$\Sigma_.$ = Alfabeto de puntuación (**(** , , , **)**)

Y las $\mapsto_i$ son reglas formadas por un par de términos ($s \mapsto t$)

Ejemplo
Un sistema de reescritura para funciones lógicas se escribiría más o menos como sigue:
({**and**, **or**, **(** , **)**, ,, **true**, **false**},
(**and**(**true**,x) ↦ x, **and**(**false**,x) ↦**false**, **or**(**true**,x) ↦ **true**, **or**(**false**,x) ↦ x)
)
Más claramente, y dando por sobreentendido el alfabeto:
and(**true**,x) ↦ x
and(**false**,x) ↦**false**
or(**true**,x) ↦ **true**
or(**false**,x) ↦ x

Se define el conjunto de los términos sobre Σ, llamado Ter(Σ), de la siguiente forma:

$\Sigma_V \subseteq Ter(\Sigma)$
$T_1, \ldots, T_n \in Ter(\Sigma) \wedge F \in \Sigma_F$ con aridad n $\Rightarrow F(T_1, \ldots, T_n) \in Ter(\Sigma)$

Este conjunto Ter(Σ) es el conjunto A de un sistema de reescritura abstracto, pero para definir las relaciones de reescritura hay que definir dos operaciones sobre Ter(Σ):

Sobre el conjunto Ter(Σ) se definen dos operaciones: Sustitución y Contexto.

Sustitución
Una sustitución es una aplicación $\sigma : Ter(\Sigma) \longrightarrow Ter(\Sigma)$ que cumple las siguientes propiedades:
$\sigma(x) \in Ter(\Sigma)$ donde $x \in \Sigma_V$
$\sigma(F(T_1, \ldots, T_n)) = F(\sigma(T_1), \ldots, \sigma(T_n))$ donde $F \in \Sigma_F$ con aridad n y $T_1, \ldots, T_n \in Ter(\Sigma)$

Ejemplo
Sea $\sigma(x)$ = **false** entonces
$\sigma($ **and**(x, **true**)) = **and**(σ(x), σ(**true**)) = **and**(**false**, **true**)

Es decir, que se sustituye cada variable por su imagen y el resto se queda igual.

Contexto
Un contexto C[ ] es un término donde hay una variable especial, llamada hueco y representada por □, que ocurre sólo una vez en él. Entonces, dado un término $T \in Ter(\Sigma)$

C[T] = σ(C[ ]) siendo σ una sustitución tal que
σ(□) = T, es decir, se sustituye el hueco por el término.
$\sigma(x) = x$ si $x \in \Sigma_V$ y x no es el hueco

Ejemplo
Sea C[ ] = **and**(□, **or**(x, **true**)), entonces
C[**false**] = σ(C[]) = σ(**and**(□, **or**(x, **true**))) = **and**(σ(□),σ(**or**(x, **true**))) =
= **and**(**false**, **or**(σ(x), σ(**true**))) = **and**(**false**, **or**(x, **true**))

Es decir, que se sustituye el hueco por el argumento del contexto y el resto se queda igual. En principio los contextos serán de un único hueco, pero se podrán hacer excepciones más adelante.

Obtención de la relación de reescritura a partir de las reglas de reescritura
Ahora estamos en condiciones de definir las relaciones $\rightarrow_i$ del sistema de reescritura abstracto a partir de las reglas $\mapsto_i$ del sistema de reescritura de términos.

$$\forall C[\,], \sigma : C[\,] \text{ contexto y } \sigma \text{ sustitución} : S \mapsto_i T \Leftrightarrow C[\sigma(S)] \rightarrow_i C[\sigma(T)]$$

Al subtérmino σ(S) se le llama expresión reducible (redex).

Ejemplo
Con las reglas anteriores obtendríamos los siguientes elementos en la relación de reescritura:

Regla	**and**(**true**,x) ↦ x	**and**(**false**,x) ↦**false**	**or**(**true**,x) ↦ **true**	**or**(**false**,x) ↦ x
Relación	and(true, false) →false	and(false, false) →false	or(true, false)→ true	or(false, false) → false
	and(true, true) →true	and(false, true) →false	or(true, true) → true	or(false, true) → true
	and(true, and(true,false)) →and(true,false)	and(false, and(true,false)) →false	or(true, and(true, false)) → true	or(false, and(true, false)) → and(true, false)
	⋮	⋮	⋮	⋮

Así pues, se construye un sistema abstracto de reescritura a partir de un sistema de reescritura de términos y se pueden analizar todas las propiedades.

Sin embargo, se pueden garantizar ciertas propiedades si se construye un sistema de reescritura de términos especial.

Sistemas ortogonales de escritura de términos

Definición
Un sistema ortogonal es un sistema de reescritura de términos donde dado cualquier término T de Ter(Σ), no hay en él un funtor implicado en dos redex distintos.

Equivalentemente se puede decir que los redex del sistema no se solapan.

Ejemplo
Un ejemplo de sistema no ortogonal sería:

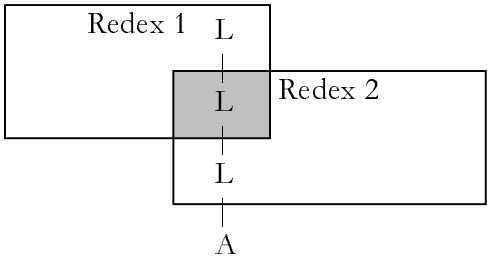
$$L(L(x)) \mapsto x$$

Ya que con el término L(L(L(A))) se podrían reducir las dos L's externas o las dos internas, estando la L intermedia implicada en las dos reducciones.

Los redex están en negrita y el funtor común subrayado en la siguiente tabla:

$$\mathbf{L(\underline{L}(}L(A)\mathbf{))} \rightarrow L(A)$$
$$L(\mathbf{\underline{L}(L(}A\mathbf{))}) \rightarrow L(A)$$

En vista arborescente sería



Ejemplo
Otro ejemplo de sistema no ortogonal sería:

$$OR(x,y) \mapsto x$$
$$OR(x,y) \mapsto y$$

Ya que con la expresión OR(A,B) se podría aplicar la regla primera o la segunda utilizando el mismo funtor OR.

Ejemplo
Un sistema ortogonal es:

$$SUMA(0,x) \mapsto x$$
$$SUMA(S(y),x) \mapsto S(SUMA(y,x))$$

Ya que no existe ningún primer argumento que satisfaga que es 0 y que es S(x).

Comprobación de ortogonalidad
Para ver si un sistema es ortogonal basta comprobar que no haya un antecedente unificable con un subtérmino (no variable) de otro antecedente (o del mismo, siendo subtérmino propio) y que cada variable no aparezca más de una vez en cada antecedente (reglas lineales por la izquierda o en los antecedentes). Esto se verá con más claridad cuando se definan los pares críticos más adelante.

Ejemplo
En $L(L(x)) \mapsto x$ puedo unificar el antecedente L(L(x)) con el subtérmino propio del mismo L(x) haciendo x=L(x) en el segundo.

Ejemplo
En OR(x,y) no hace falta ni renombrar variables ya que el antecedente de la primera regla es igual al antecedente de la segunda, es igual pero no el mismo (están en distintas reglas).

Propiedades de los sistemas ortogonales
Hay una importante propiedad: Todos los sistemas ortogonales son confluentes.

Otra propiedad es: Si en cada regla las mismas variables ocurren en antecedente y en consecuente, decimos que es un sistema no borrador, ocurre que $WN \Leftrightarrow SN$.

Si en un sistema ortogonal siempre se puede encontrar una forma normal mediante el orden de evaluación aplicativo (más interior), entonces es SN y viceversa.

Estrategias de reducción en los sistemas ortogonales

Clasificación de redexes

Notaremos que un redex s está incluido en otro redex r de la siguiente manera: $s \subset r$, y lo definimos así

$$s \subset r \Leftrightarrow \exists D[] : D[s] = r$$

Diremos que un redex r que está en una expresión T ($r \subset T$) es un redex más interior si y sólo si $\forall s \subset T : s \not\subset r$

Diremos que un redex r que está en una expresión T ($r \subset T$) es un redex más exterior si y sólo si $\forall\ s \subset T : r \not\subset s$

Estrategias de reducción

Tenemos cinco estrategias principales según los redex que vayamos a reducir. En la siguiente tabla están listadas:

Nombres	Es normal	Es cofinal	Es concurrente
Más a la izquierda, más interior Aplicativo	No	No	No
Todos los interiores	No	No	Si
Más a la izquierda, más exterior	No (Sí en sistemas ortogonales y normales por la izquierda)	No	No
Todos los exteriores	Si	No	Si
Todos los redexes (Sólo válida en sistemas ortogonales) Reducción de Kleene Reducción de Gross—Knuth	Si	Si	Si

Las propiedades mencionadas son:

- Normal: Siempre llega a una forma normal y, como es confluente, es única.
- Cofinal: Toma el camino más corto para llegar a esa forma normal.
- Concurrente: Reduce varios redexes en un paso.

La reducción más a la izquierda, más exterior es normal en los sistemas ortogonales normales por la izquierda. En estos sistemas no hay constantes ni funtores a la derecha de una variable.

Técnicas para estudiar la terminación. Método de Dershowitz

Se ha visto anteriormente que se cumple que WCR & SN → CR , o sea, dado un sistema de reescritura débilmente confluente, nos bastaría comprobar que termina para asegurar que es confluente (CR). En vista a lo anterior la terminación parece una propiedad bastante útil. Pero el determinar si un sistema de reescritura de términos genera o no derivaciones infinitas es un problema indecidible (no existe ningún algoritmo que en tiempo finito me responda bien "SI TERMINA", o bien "NO TERMINA"). Pero bajo ciertas condiciones son aplicables técnicas que nos pueden ayudar a discernir sobre la terminación de algunos TRS's. La que nosotros vamos a explicar está basada en un teorema de J. B. Kruskal y fue desarrollada por N. Dershowitz en 1987.

El objetivo del método consiste en conseguir expresar cada una de las reglas de un TRS T como un caso particular de una TRS especial ($\Rightarrow^+$) que cumple la propiedad SN. De esta forma estamos seguros que al aplicar el TRS T , nunca se producirán derivaciones infinitas. Dicha TRS $\Rightarrow^+$ ha de ser suficientemente potente como para cubrir una gran variedad de casos, y no sólo eso, también será necesario demostrar que $\Rightarrow^+$ cumple SN, con la ventaja que para cualquier otro TRS's nos bastaría expresarlo en función de $\Rightarrow^+$ para aseguramos su terminación. La demostración de la terminación de $\Rightarrow^+$ está basada en el teorema de Kruskal, que a su vez tiene una demostración de extrema complejidad.

Vamos a definir $\Rightarrow^+$ sobre un conjunto de árboles $\mathcal{A}$ (árbololes de nodos etiquetados con números naturales), por lo cada regla ha de ser estudiada como una transformación de un árbol de $\mathcal{A}$ en otro árbol de $\mathcal{A}$. Empezaremos definiendo formalmente el conjunto $\mathcal{A}$. Luego definiremos un superconjunto de $\mathcal{A}$, que llamaremos $\mathcal{A}^*$, y definiremos sobre $\mathcal{A}^*$ una relación $\Rightarrow$. Veremos que $\Rightarrow^+$ (cierre transitivo de $\Rightarrow$) restringida a $\mathcal{A}$ cumple la propiedad SN. Hecho esto, el método en cuestión consiste en, dado un TRS T, buscar una manera asignar a los funtores un peso (que va a ser un número natural), de forma que cada regla de T sea una función de $\mathcal{A}$ en $\mathcal{A}$. Después se debe expresar cada regla de T como instancia de $\Rightarrow^+$. El éxito del método en ese caso concreto residirá en nuestra astucia a la hora de establecer la asignación de pesos.

El conjunto $\mathcal{A}$

Definición 1
Un árbol $a \in \mathcal{A}$ es un par ($< \mathcal{D}, \leq, \alpha >$, $\mathcal{L}$), donde $\mathcal{D}$ es un conjunto finito que llamaremos nodos, $\alpha \in \mathcal{D}$ es un nodo destacado al que llamaremos raíz, y $\leq$ es una relación de orden parcial sobre $\mathcal{D}$.
Se exige que:

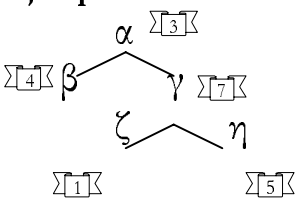
1. $\forall\, \delta \in \mathcal{D} : \delta \leq \alpha$
2. $\forall\, \delta, \psi, \xi \in \mathcal{D}: (\delta \leq \psi \ \wedge \delta \leq \xi) \Rightarrow (\xi \leq \psi \vee \psi \leq \xi)$

Por otra parte, $\mathcal{L}: \mathcal{D} \rightarrow \mathbb{N}$, es una función que asigna a cada nodo un número natural, es decir, etiqueta cada nodo con un peso. Diremos también que $\mathcal{D}$ es igual $\mathcal{N}(a)$ (conjunto de nodos del árbol a).

Cuando $\psi \leq \xi$ y son distintos decimos que ψ es un descendiente de ξ, y que ξ es un ascendiente de ψ . Además, si para cualquier δ distinto a ψ que verifique $\psi \leq \delta$ se cumple $\xi \leq \delta$, entonces diremos que ψ es un hijo de ξ y que ψ es el padre ξ. Eso se refleja en las figuras porque ψ y ξ están unidos con una línea.

La primera exigencia de la definición se traduce a que la raíz de un árbol es ascendiente del resto de los nodos; la segunda significa que si un nodo tiene dos ascendientes, uno de ellos es ascendiente del otro, de lo que se infiere que dados dos nodos distintos entre los que no hay relación de ascendencia, no puede haber descendientes comunes a ambos.

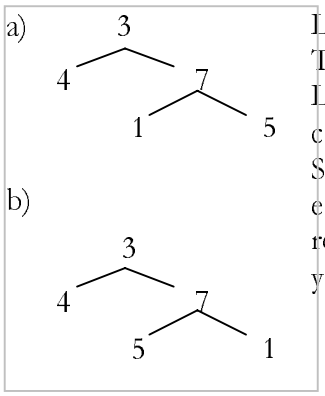
Ejemplo 1



Aquí tenemos un árbol *a* definido como :

$\mathcal{D} = \{ \alpha, \beta, \gamma, \zeta, \eta \}$
$\leq = \{ (\zeta,\zeta),(\zeta,\gamma),(\zeta,\alpha),(\eta,\eta),(\eta,\gamma),(\eta,\alpha),(\gamma,\gamma),(\gamma,\alpha),(\beta,\beta),(\beta,\alpha),(\alpha,\alpha)\}$
$\mathcal{L}(\alpha)=3 ; \mathcal{L}(\beta)=4 ; \mathcal{L}(\gamma)=7 ; \mathcal{L}(\zeta)=1 ; \mathcal{L}(\eta)=5$

A partir de ahora para designar un nodo usaremos en lugar de su verdadero nombre α, β, γ... su correspondiente etiqueta. Entonces el árbol de la figura anterior quedaría como puede verse en la figuras de abajo.



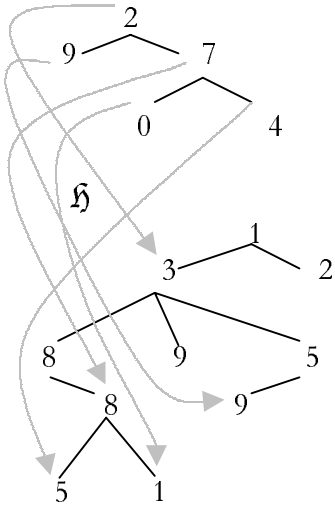
La figura a) es una representación gráfica del árbol *a* de arriba. También podría expresarse usando un término como 3(4,7(1,5)). La figura b) es otra representación del árbol *a*, que en consonancia con lo anterior podríamos escribir como 3(4,7(5,1)). Sucede que según la definición de árbol que se ha dado, no se exige ninguna ordenación entre los hijos de un mismo padre, resultando que nuestros árboles son conmutativos. La figura a) y b) son el mismo árbol.

Definición 2

Sean $a, b \in \mathcal{A}$. Diremos que $a \preccurlyeq b$ (*a* está "embebido" en *b*) si existe una función $\mathfrak{H} : \mathcal{N}(a) \rightarrow \mathcal{N}(b)$ tal que:

1. $\mathfrak{H}$ es inyectiva [$\mathfrak{H}(\phi) = \mathfrak{H}(\psi) \Rightarrow \phi = \psi$]
2. $\mathfrak{H}$ conserva el supremo [$\mathfrak{H}(\sup\{\phi, \psi\}) = \sup\{\mathfrak{H}(\phi), \mathfrak{H}(\psi)\}$]
3. $\mathfrak{H}$ incrementa la etiqueta [$\mathcal{L}_a(\phi) \leqq \mathcal{L}_b(\mathfrak{H}(\phi))$], donde $\mathcal{L}_a$, $\mathcal{L}_b$ son las funciones de etiquetado de *a* y *b* respectivamente, y $\leqq$ es la relación de orden sobre los números naturales.

Ejemplo 2

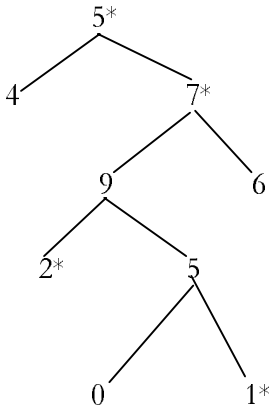


El árbol de arriba se puede representar como 2(9,7(0,4)) y el de abajo como 1(3(8(8(5,1)),9,5(9)),2)
Entre los nodos del primer y del segundo árbol se ha establecido una función $\mathfrak{H}$ tal como aparece en la figura. Como tal función $\mathfrak{H}$ cumple las tres condiciones antes mencionadas, entonces podemos afirmar que el primero está embebido en el segundo, o sea,

$$2(9,7(0,4)) \preccurlyeq 1(3(8(8(5,1)),9,5(9)),2)$$

La relación $\preccurlyeq$ es de orden parcial sobre $\mathcal{A}$

Ejemplo 5 de árbol $\mathscr{A}^*$



Definición 3

Sea $\mathscr{A}^*$ otro conjunto de árboles finitos y conmutativos, con nodos etiquetados también con números naturales, con la salvedad de que algunos de estos nodos pueden estar macados con un asterisco (*). Es fácil entender $\mathscr{A} \subseteq \mathscr{A}^*$.

(la definición formal de $\mathscr{A}^*$ no difiere de la de $\mathscr{A}$, salvo la función de etiquedado que en este caso es diferente: $\mathscr{L}^* : \mathscr{D} \rightarrow (\mathbb{N},\text{Bool})$, de forma que a un nodo se le asigna (n,True) si está marcado con *, mientras que se le asigna (n,False) si no está marcado.)

Definición 4

Definamos sobre $\mathscr{A}^*$ la relación de reducción $\Rightarrow$ como:

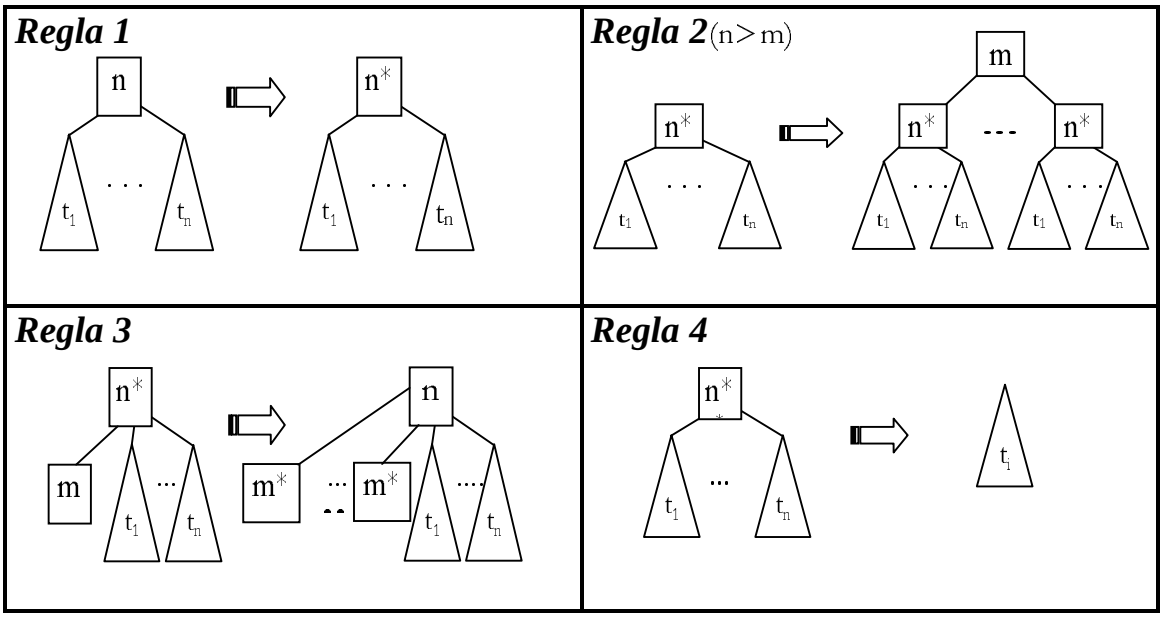
Siendo n,m naturales y t_i cualquier árbol,

Regla 1. $n(t_1,t_2,...,t_n) \Rightarrow n^*(t_1,t_2,...,t_n)$

Regla 2. Si $n>m$, $n^*(t_1,t_2,...,t_n) \Rightarrow m(n^*(t_1,t_2,...,t_n),n^*(t_1,t_2,...,t_n),...,n^*(t_1,t_2,...,t_n))$

Regla 3. $n^*(m,t_1,...,t_n) \Rightarrow n(m^*, m^*,...,m^*_1,t_1,...,t_n)$, donde m^* aparece repetida j veces con $j \geqq 0$.

Regla 4. $n^*(t_1, t_2,...,t_n) \Rightarrow t_i$ donde $i \in \{1,...,n\}$ con $n \geqq 1$.



Teorema 1

La relación $\Rightarrow^+$ restringida a $\mathscr{A}$ cumple las siguientes propiedades:

i) Para ningún $t \in \mathscr{A}$ se cumple $t \Rightarrow^+ t$ (irreflexiva).

ii) Para ningún $t_1, t_2 \in \mathscr{A}$ se cumple $t_1 \Rightarrow^+ t_2 \Rightarrow^+ t_1$ (antisimétrica)

iii) Si $t_1 \Rightarrow^+ t_2$ y $t_2 \Rightarrow^+ t_3$, entonces $t_1 \Rightarrow^+ t_3$ (transitiva).

Entonces $\Rightarrow^+$ es una relación de orden parcial estricto.

Teorema 2
Sean s,t $\in \mathscr{A}^*$

Si $s \preccurlyeq t$ (s está embebido en t), entonces $t \Rightarrow^* s$

Teorema 3. (Teorema de Kruskal de los árboles)

Sea a_0 , a_1 , a_2 , ... una sucesión de árboles de $\mathscr{A}$. Entonces existen i , $j \in \mathbb{N}$, con $i < j$ tal que $a_i \preccurlyeq a_j$.

Teorema 4
La relación $\Rightarrow^+$, restringida a $\mathscr{A}$ cumple *SN*. Esto significa que no existe ninguna secuencia infinita $t_0 \Rightarrow^+ t_1 \Rightarrow^+ t_2 \Rightarrow^+$ ... siendo los t_i árboles sin marcas.

Supongamos que existe una secuencia infinita ...

$$t_0 \Rightarrow^+ t_1 \Rightarrow^+ t_2 \Rightarrow^+ \ldots \Rightarrow^+ t_i \Rightarrow^+ \ldots \Rightarrow^+ t_j \Rightarrow^+ \ldots$$

Por el teorema de Kruskal, habrá dos i, j $\in \mathbb{N}$ con i < j tal que $t_i \preccurlyeq t_j$, y entonces por el teorema 2 tenemos que $t_j \Rightarrow^* t_i$. Ahora bien, por la misma secuencia tengo que $t_i \Rightarrow^+ t_j$ y también veo que tengo $t_j \Rightarrow^* t_i$, con lo que finalmente tengo $t_i \Rightarrow^+ t_i$, y esto entra en contradicción con el teorema 1, donde se decía que $\Rightarrow^+$ era irreflexiva.

Método de Dershowitz

Sea *R* un sistema de reescritura de términos. Primero tenemos que asignar un peso a cada uno de los funtores que intervienen en las reglas (de lo afortunada que sea esta asignación dependerá el éxito del método). Si sustituimos los nodos por sus pesos, cada antecedente de las reglas del sistema será un elemento de $\mathscr{A}$ (idem para cada consecuente). Luego, regla por regla, trataremos de reducir el antecedente de la misma hasta el consecuente que le corresponda usando exclusivamente $\Rightarrow^+$. Si esto es posible para cada regla habremos demostrado que cada una de ellas es un caso particular de $\Rightarrow^+$, y como $\Rightarrow^+$ es ***SN*** cuando nos restringimos a $\mathscr{A}$, nuestro *TRS* también será ***SN.***

Ejemplo 6:
Sea *R un TRS* de como el de la figura

$$\begin{array}{lcl} \neg\neg x & \rightarrow & x \\ \neg(x \vee y) & \rightarrow & (\neg x \wedge \neg y) \\ \neg(x \wedge y) & \rightarrow & (\neg x \vee \neg y) \\ x \wedge (y \vee z) & \rightarrow & (x \wedge y) \vee (x \wedge z) \\ (y \vee z) \wedge x & \rightarrow & (y \wedge x) \vee (z \wedge x) \end{array}$$

Primero asignamos pesos:

Asignación de pesos:

$$\begin{array}{lcl} \vee & - & 1 \\ \wedge & - & 2 \\ \neg & - & 3 \end{array}$$

Para la primera regla :

- Primero pasamos el antecedente de la regla a la notación arbórea, con los pesos numéricos:
  $\neg\neg x$ se escrbiría como 3(3(x))
- Luego hacemos sucesivas transformaciones al árbol inicial hasta convertirlo en el que le corresponde a la parte derecha de la regla.

$$\begin{aligned} 3(3(x)) &\Rightarrow 3^*(3(x)) \\ &\Rightarrow 3(x) \\ &\Rightarrow 3^*(x) \\ &\Rightarrow x \end{aligned}$$

- Finalmente, es fácil dase cuenta que $3(3(x)) \Rightarrow^+ x$, y si volvemos a escribir en notación estandar tenemos $\neg\neg x \Rightarrow^+ x$. Esta regla es una aplicación particular de $\Rightarrow^+$, de la que ya tenemos asegurada la terminación.

Para la segunda regla es parecido,

$$\begin{aligned} 3(1(x,y)) &\Rightarrow 3^*(1(x,y)) \\ &\Rightarrow 2(3^*(1(x,y)) , 3^*(1(x,y))) \\ &\Rightarrow 2(3(1^*(x,y)), 3(1^*(x,y))) \\ &\Rightarrow 2(3(x), 3(y)) \end{aligned}$$

de forma que tenemos $3(1(x,y)) \Rightarrow^+ 2(3(x), 3(y))$, que pasado a notación estándar es :

$$\neg(x \vee y) \rightarrow (\neg x \wedge \neg y)$$

Se procede análogamente para el resto de las reglas.

Transformación de sistemas ecuacionales en TRS's

Especificación de Sistemas Ecuacionales

Un sistemas ecuacional es algo muy parecido a un TRS, con la diferencia que en un sistema ecuacional no existe sentido de aplicación de las reglas, sino que las reglas, que aquí son ecuaciones, se pueden leer en ambos sentidos, de izquierda a derecha y de derecha a izquierda. Y digo leerlas, ya que en un sistema ecuacional, más que dar un mecanismo de transformación, lo que se pretende es dar un conjunto de igualdades que se verifican en un determinado dominio que pretendemos describir.

Formalmente un Sistema Ecuacional (a partir de ahora ***SE***) es un par (Σ, E), donde Σ tiene el mismo significado que tenía en los TRS's , y E es un conjunto de ecuaciones de la forma $s=t$ donde $s,t \in Ter(\Sigma)$.

Es muy posible, que en un sistema de ecuaciones se digan más cosas de las que explícitamente aparecen, ya que a partir de un conjunto inicial de ecuaciones E se pueden deducir otras ecuaciones, usando para ello un sistema de inferencia como el descrito a continuación:

Definición 5 (Sistema Axiomático)
Axiomas

- Si $t=s \in E$ entonces $(\Sigma, E) \vdash t=s$ *(los elementos de **E** son axiomas)*
- $(\Sigma, E) \vdash t=t$ *(reflexividad)*

Reglas de Inferencia

Simetría

$$\frac{(\Sigma, E) \vdash s=t}{(\Sigma, E) \vdash t=s}$$

Sustitución (1)

$$\frac{(\Sigma, E) \vdash s=t}{(\Sigma, E) \vdash s^{\sigma}= t^{\sigma}} \quad \textit{(para cualquier sustitución } \sigma \textit{)}$$

Sustitución (1)

$$\frac{(\Sigma, E) \vdash s_1=t_1,\ldots,(\Sigma, E) \vdash s_n=t_n}{(\Sigma, E) \vdash F(s_1,\ldots,s_n)= F(t_1,\ldots,t_n)}$$

Transitividad

$$\frac{(\Sigma, E) \vdash t_1=t_2 , (\Sigma, E) \vdash t_1=t_2}{(\Sigma, E) \vdash t_1=t_3}$$

Definición 6. Corrección de un TRS con respecto a un SE

Diremos que un TRS **R** es correcto con respecto a un sistema ecuacional (Σ, E) si la igualdad inducida por el TRS **R** implica igualdad según el sistema ecuacional, o sea,

$$\forall\ t,s \in Term(\Sigma) \qquad t =_{\mathbf{R}} s \ \Rightarrow (\Sigma, E) \vdash t = s$$

Estudiar si un TRS concreto es correcto o no lo es, es bastante fácil. Bastaría con tomar regla por regla, y hacer lo siguiente:

- Reducir t a su forma normal t '
- Reducir s a su forma normal s '
- Comparar t ' con s ' , si son idénticos pasamos a la siguiente regla, si no son idénticos el TRS no es correcto.
- Cuando terminemos con éxito todas las reglas habremos terminado. El TRS será correcto con respecto al sistema ecuacional.

NOTA IMPORTANTE:

El método para estudiar la corrección de un TRS dado arriba sólo funciona si el TRS cumple las propiedades CR y SN, o sea, es confluente y termina. De lo contrario nos podemos quedar "colgados" intentando pasar t o s a sus respectivas formas normales.

Definición 7. Completitud de un TRS con respecto a un SE

Diremos que un TRS **R** es completo con respecto a un sistema ecuacional (Σ, E) si la igualdad definida por el sistema ecuacional (teniendo en cuenta el sistema de inferencia antes presentado) implica igualdad según el TRS , o sea,

$$\forall\ t, s \in Term(\Sigma) \qquad (\Sigma, E) \vdash t = s \ \Rightarrow t =_R s$$

Dado un conjunto de ecuaciones de la forma $f_1 = g_1, f_2 = g_2, f_3 = g_3, \ldots, f_n = g_n$, sería interesante obtener a partir de él un sistema de reescritura que fuera correcto y completo con respecto al conjunto de ecuaciones, y no sólo eso, sino que también cumpla ***CR*** y ***SN***

Cada ecuación de la forma $f_i = g_i$ plantea un dilema a la hora de obtener una regla, ya que esa regla podría ser bien $f_i \rightarrow g_i$, o bien $g_i \rightarrow f_i$. A voz de pronto, a alguien se le puede ocurrir incluir en el TRS que queremos construir ambas reglas. Pero en ese caso el TRS no cumpliría ***SN*** ya que podríamos aplicar simultáneamente una y la otra y obtendríamos una derivación infinita, cosa que no nos interesa.

¿Cuál de las dos reglas posibles incorporamos a nuestro *TRS* ? A veces una de las opciones queda descartada debido a que no cumple las condiciones sintácticas que los *TRS's* imponen a sus reglas.

Ejemplo 7 Si tenemos una ecuación como $x+0=x$, se nos plantean las reglas $x+0 \rightarrow x$ y $x \rightarrow x+0$. Pero la segunda de las reglas queda decartada, ya que ninguna regla puede tener una parte izquierda igual a una variables.

Pero otras veces las dos opciones cumplen las condiciones para ser reglas. En estos casos no está claro el sentido de la reducción. La elección está en nuestras manos y determinará las propiedades del sistema de reescritura que obtengamos. De cualquier manera, al eliminar uno de los dos sentidos de la ecuación, perdemos información, que de alguna manera hay que rescatar para intentar tener un sistema lo más cercano a la completitud que se pueda, ya que la completitud es difícilmente alcanzable.

Definición 8: Par Crítico
Sea *R* un *TRS*, y sean $\alpha \mapsto \beta$ y $\gamma \mapsto \delta$ dos reglas de *R* tal que α es unificable con un subtérmino no-variable de γ. Sea σ el unificador más general de ambos y sea *t* un término tal que $\sigma(t) \equiv \sigma(\alpha)$. Entonces el término $\sigma(\gamma)$ sería de la forma $\sigma(C[t])$ puede reducirse de dos maneras distintas, usando la primera regla, o bien usando la segunda. Dando dos posibles reducciones, por un lado $C[\sigma(\beta)]$ y por otro $\sigma(\delta)$. A este par de valores $<C[\sigma(\beta)],\sigma(\delta)>$ se le llama **par crítico**.

La confluencia del sistema dependerá de la capacidad de reducción de ambos términos a uno común. Un par crítico va asociado a lo que en apartados anteriores llamábamos solapamiento de reglas

Definición 9: Convergencia
De un par crítico $<s, t>$ se dice que es convergente en un *TRS* (y se escribe $s \downarrow t$) si es posible una reducción a un término común.

Ejemplo 8

Sea el siguiente conjunto de ecuaciones:

$$e \circ x = x$$

$$I(x) \circ x = e$$

$$(x \circ y) \circ z = x \circ (y \circ z)$$

Entonces podemos obtener:

- De la primera ecuación obtenemos la regla
  1. $e \circ x \rightarrow x$
  (se descarta el sentido contrario, porque habría una variable sóla en la parte izquierda)
- De la segunda ecuación obtenemos la regla
  2. $I(x) \circ x \rightarrow e$
  (se descarta el sentido contrario, porque habría variable en la parte derecha que no han aparecido previamente en la parte izquierda)
- De la tercera ecuación obtenemos la regla
  3. $(x \circ y) \circ z \rightarrow x \circ (y \circ z)$
  (no hay motivos para decartar la otra opción, pero no podemos quedarnos con las dos)

De forma que hemos obtenido un TRS inicial:

1. $e \circ x \rightarrow x$
2. $I(x) \circ x \rightarrow e$
3. $(x \circ y) \circ z \rightarrow x \circ (y \circ z)$

- Ahora podemos ver que tenemos problemas:
  Si tengo el término (I(x) ∘ x) ∘ z ; por un lado puedo aplicarle la regla 2 y luego la 1, de forma que se transformaría en z ; por otro podría aplicar la regla 3, de manera que llegaríamos a I(x) ∘ (x ∘ z). Según el sistema ecuacional z debería se igual a I(x) ∘ (x ∘ z), pero según el TRS no lo son, pues no hay forma de hacerlos confluir a la misma forma normal. Para solventar este escollo añadimos explícitamente la regla
  4. $I(x) \circ (x \circ z) \rightarrow z$

- Pero de nuevo hay problemas, pues con la nueva regla introducida tenemos que el término I(I(y)) ∘ (I(y) ∘ y), por la regla 2 se convierte en I(I(y)) ∘ e , pero por la regla 4 se convierte en y. De forma que <y,I(I(e))> forman un par crítico que no es convergente, así que, lo hacemos convergente explícitamente con la nueva regla:
  5. $I(I(y)) \circ e \rightarrow y$

- Además, tenemos otro par crítico; cuando el término a reducir sea I(e) ∘ (e ∘ z) podemos aplicar la regla 1, y obtenemos I(e) ∘ z ; o bien aplicar la regla 4, y entonces tenemos z. Luego tenemos que añadir la regla
  6. $I(e) \circ z \rightarrow z$

- También hay superposición entre las reglas 3 y 5: $(I(I(y)) \circ e) \circ x \rightarrow_3 I(I(y)) \circ (e \circ x)$ $(I(I(y)) \circ e) \circ x \rightarrow_5 y \circ x$. Como además $(e \circ x) \rightarrow_1 x$, tengo que añadir la regla
  7. $I(I(y)) \circ x \rightarrow y \circ x$

- Hay de nuevo superposición entre la reglas 5 y 7: $I(I(y)) \circ e \rightarrow_5 y$,por otro lado tenemos que $I(I(y)) \circ e \rightarrow_7 y \circ e$; el problema se resulve añadiendo
  8. $y \circ e \rightarrow y$

- Por superposición entre la reglas 5 y 8: $I(I(y)) \circ e \rightarrow_5 y$,por otro lado tenemos que $I(I(y)) \circ e \rightarrow_8 I(I(y))$; el problema se resulve añadiendo
  9. $I(I(y)) \rightarrow y$

- Con todo lo que hemos añadido hasta ahora resulta que la regla 5 ahora no es necesaria porque ya se puede poner como composición de las reglas 9 y 8, o sea, lo que antes se hacía con la 5 [$I(I(y)) \circ e \rightarrow_5 y$] ahora se hace con 9 y la 8:
  $I(I(y)) \circ e \rightarrow_9 y \circ e \rightarrow_8 y$ (eliminamos la 5)

- Por la misma razón que antes resulta que la siete no es necesaria, ya que puede ser suplida mediante la regla 9,lo que antes se hacía con la 7 [$I(I(y)) \circ x \rightarrow_7 y \circ x$] ahora se hace con 9 :
  $I(I(y)) \circ x \rightarrow_9 y \circ x$ (eliminamos la 7)

- Por superposición entre la reglas 6 y 8: $I(e) \circ e \rightarrow_6 e$,por otro lado tenemos que $I(e) \circ e \rightarrow_8 I(e)$; el problema se resulve añadiendo
  10. $I(e) \rightarrow e$

- Con todo lo que hemos añadido hasta ahora resulta que la regla 6 ahora no es necesaria porque ya se puede poner como composición de las reglas 10 y 1, o sea, lo que antes se hacía con la 6 [$I(e) \circ z \rightarrow_6 z$] ahora se hace con 10 y la 1:
  $I(e) \circ z \rightarrow_{10} e \circ z \rightarrow_1 z$ (eliminamos la 6)
- Por superposición entre la reglas 2 y 9: $I(I(y)) \circ I(y) \rightarrow_2 e$,por otro lado tenemos que $I(I(y)) \circ I(y) \rightarrow_9 y \circ I(y)$; el problema se resulve añadiendo
  11. $y \circ I(y) \rightarrow e$

- Por superposición entre la reglas 3 y 11: $(y \circ I(y)) \circ x \rightarrow_3 y \circ (I(y) \circ x)$,por otro lado tenemos que $(y \circ I(y)) \circ x \rightarrow_{11} y \circ x$; el problema se resulve añadiendo
  12. $y \circ (I(y) \circ x) \rightarrow y \circ x$

- Por superposición entre la reglas 3 y 11 (otra vez): $(x \circ y) \circ I(x \circ y) \rightarrow_3 x \circ (y \circ I(x \circ y))$, por otro lado tenemos que $(x \circ y) \circ I(x \circ y) \rightarrow_{11} e$; el problema se resulve añadiendo
  13. $x \circ (y \circ I(x \circ y)) \rightarrow e$

- Por superposición entre la reglas 4 y 13 : $I(x) \circ (x \circ (y \circ I(x \circ y))) \rightarrow_4 y \circ I(x \circ y)$, por otro lado tenemos que $I(x) \circ (x \circ (y \circ I(x \circ y))) \rightarrow_{13} I(x) \circ e \rightarrow_8 I(x)$; el problema se resulve añadiendo
  14. $y \circ I(x \circ y) \rightarrow I(x)$
- Con todo lo que hemos añadido hasta ahora resulta que la regla 13 ahora no es necesaria porque ya se puede poner como composición de las reglas 14 y 11, o sea, lo que antes se hacía con la 13 [$x \circ (y \circ I(x \circ y)) \rightarrow_{13} e$] ahora se hace con 14 y la 11:
  $x \circ (y \circ I(x \circ y)) \rightarrow_{14} x \circ I(x) \rightarrow_{11} e$ (eliminamos la 13)

- Por superposición entre la reglas 4 y 14 : $I(y) \circ (y \circ I(x \circ y)) \rightarrow_4 I(x \circ y)$, por otro lado tenemos que $I(y) \circ (y \circ I(x \circ y)) \rightarrow_{14} I(y) \circ I(x)$; el problema se resulve añadiendo
  15. $I(x \circ y) \rightarrow I(y) \circ I(x)$

- Con todo lo que hemos añadido hasta ahora resulta que la regla 14 ahora no es necesaria porque ya se puede poner como composición de las reglas 15 y 12, o sea, lo que antes se hacía con la 14 [$y \circ I(x \circ y) \rightarrow_{14} I(x)$] ahora se hace con 15 y la 12:
  $y \circ I(x \circ y) \rightarrow_{15} y \circ (I(y) \circ I(x)) \rightarrow_{12} I(x)$ (eliminamos la 14)

Finalmente obtenemos:

1.	$e \circ x$	$\rightarrow$	x
2.	$I(x) \circ x$	$\rightarrow$	e
3.	$(x \circ y) \circ z$	$\rightarrow$	$x \circ (y \circ z)$
4.	$I(x) \circ (x \circ z)$	$\rightarrow$	z
8.	$y \circ e$	$\rightarrow$	y
9.	$I(I(y))$	$\rightarrow$	y
10.	$I(e)$	$\rightarrow$	e
11.	$y \circ I(y)$	$\rightarrow$	e
12.	$y \circ (I(y) \circ x)$	$\rightarrow$	$y \circ x$
15.	$I(x \circ y)$	$\rightarrow$	$I(y) \circ I(x)$

Teorema 5 Lema del par crítico:
Sea *R* un *TRS*, ***WCR***(*R*) $\Leftrightarrow$ ($\forall <s , t>$: par crítico de *R* : $s \downarrow t$)

Lo que se quiere decir es que si queremos que un TRS sea débilmente confluente, todo par crítico del TRS debe ser convergente, o sea, debe haber alguna forma de hacer que los dos elementos del par crítico llegen a la misma expresión.

Teorema 6. Corolario: Sea *R* un *TRS*, tal que ***SN***(*R*). Entonces:
CR(*R*) $\Leftrightarrow$ ($\forall <s , t>$: par crítico de *R* : $s \downarrow t$)

Si además nos aseguramos la terminación, entonces será confluente.

Definición10 : Orden de reducción
Un orden de reducción > es un orden parcial sobre los el conjunto de términos que cumple:

$s > t \Rightarrow \sigma(s) > \sigma(t)$
$s > t \Rightarrow C[s] > C[t]$

Con esto se da un criterio de reducción, que me permite, en caso de duda saber en que sentido hay que escribir una reducción (como ocurría en el ejemplo 7.

Algoritmo de Knuth-Bendix (para un orden de reducción prefijado)
Sea el E el conjunto de ecuaciones y sea R el cojunto de reglas.

```
R:=Ø;
     mientras E tenga elementos
          seleccionar una ecuación s=t de E
          reducir s → s̄  y t → t̄
          si s̄ ≡ t̄ entonces
               E:= E - { s=t }
          en otro caso
               si (s̄ > t̄) entonces
                    α := s̄ ;  β := t̄
               en caso de que s̄ > t̄
                    β := s̄ ;  α := t̄
               en otro caso
                    fracaso
               fin_si

               CP:={ P=Q | <P,Q> es un par crítico,
                    entre las reglas de R y α→β }
               R:=R ∪{α→β}
               E:=E ∪ (CP-[s=t ])
          fin_si
     fin_mientras
```

Epílogo

Con los conocimientos teóricos recién adquiridos, podemos volver a los ejemplos mencionados en la introducción para llegar a interesantes conclusiones.

Ecuaciones Matématicas

Hemos visto un algoritmo para transformar ecuaciones matemáticas en reglas de reescritura. Dicho algoritmo no es determinista y no funciona en todos los casos, pero a cambio los sistemas de reescritura generados son ortogonales y por tanto consistentes (si hay dos formas normales distintas).

Haskell

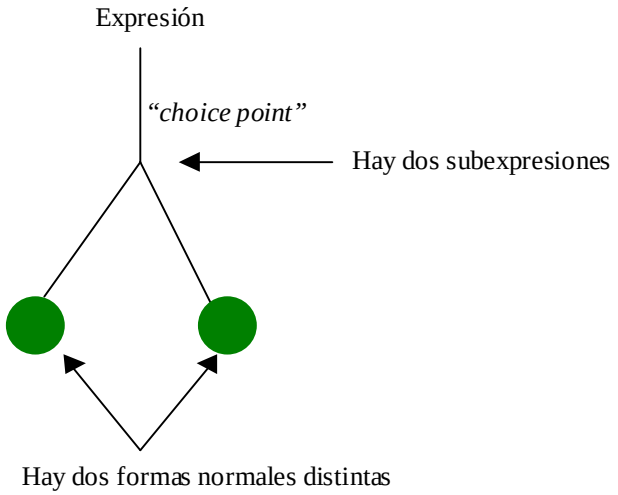
Haskell es un sistema casi ortogonal, ya que aunque el sistema permite definir dos reglas que se solapen, solo la primera que aparezca en el código será la efectiva, de manera que no se pueden aplicar las dos a la vez y no se pueden producir pares críticos. Esto hace que Haskell sea un sistema confluente y por tanto consistente. Por otro lado, esto no implica que Haskell tenga terminación asegurada, basta con escribir una regla del tipo:

```
f x = f x
```

Prolog

Como ya se dijo, Prolog es muy distinto a los dos casos anteriores aun asi se pueden llegar a algunas conclusiones.

Por ejemplo, si dos reglas se solapan (si las dos se pueden aplicar sobre un mismo redex) el Prolog introduce un "Choice-Point", de manera que nunca se van aplicar dos reglas en una misma rama. Siendo cada rama ortogonal y por lo tanto consistente. Por otro lado, gracias al backtracking tenemos varias formas normales, y cada una de ellas será un resultado de la computación.



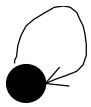
λ-Cálculo

El λ-Cálculo ya ha sido estudiado, pero podemos destacar el teorema de Church- Rosser que dice que es confluente y por tanto si tiene dos formas normales distintas (que las tiene, basta tomar dos variables distintas) es consistente.

Dado que el λ-calculo es un modelo de computación, cualquier otro modelo de computación será equivalente. En particular las maquinas de Turing

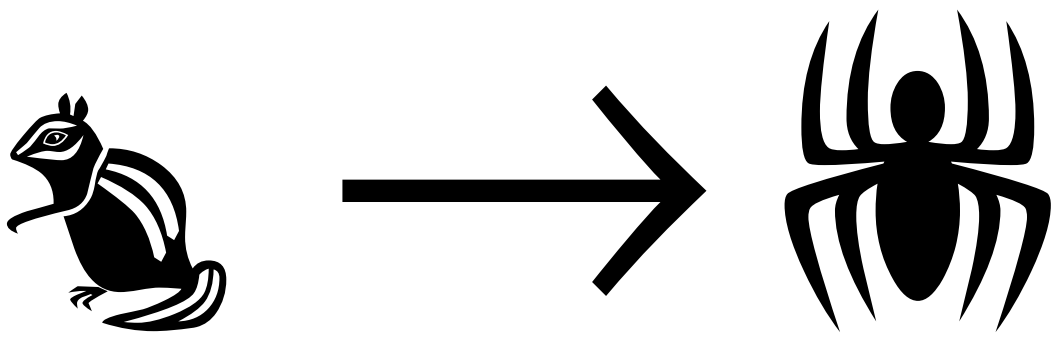
Máquina de Turing

Si tomamos como expresión su configuración obtenemos unos sistemas de reescritura de configuraciones basado en la tabla de estados de la maquina, que harían la función de reglas. Hay que hacer notar que tampoco se garantiza la terminación, ya que podemos tener el siguiente autómata:



Apéndice: Transparencias usadas en la presentación

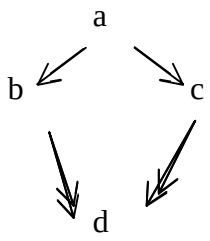
Sistemas de Reescritura



PROPIEDADES

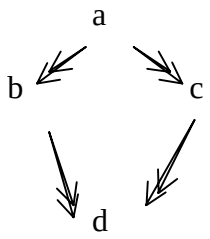
Confluencia Débil WCR()

à, b, c : a, b, c ∈ A : ∃d : d ∈ A



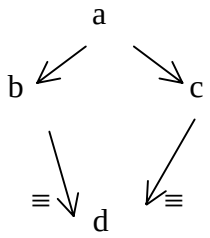
Confluencia CR()

à, b, c : a, b, c ∈ A : ∃d : d ∈ A



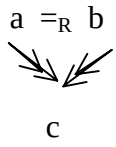
Subconmutativa WCR≤1

à, b, c : a, b, c ∈ A : ∃d : d ∈ A



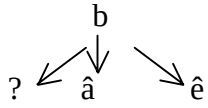
Confluencia por Equivalencia CEq(=R)

à, b : a, b ∈ A : ∃c : c ∈ A



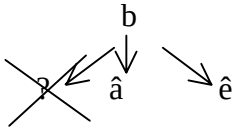
Débilmente Normalizante WN().

∀ b : b ∈ A : ∃ â : â ∈ A : â →b



Fuertemente Normalizante SN()

Es WN →Terminación



Únicamente Normalizante UN(=R)

∀ â, ê : â, ê ∈ A : â=Rê →â=ê

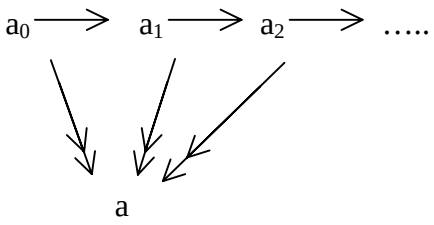
Incremental Inc()

∃ f : f:A →N : ∀ a, b : a, b ∈ A : a→b →f(a) < f(b)

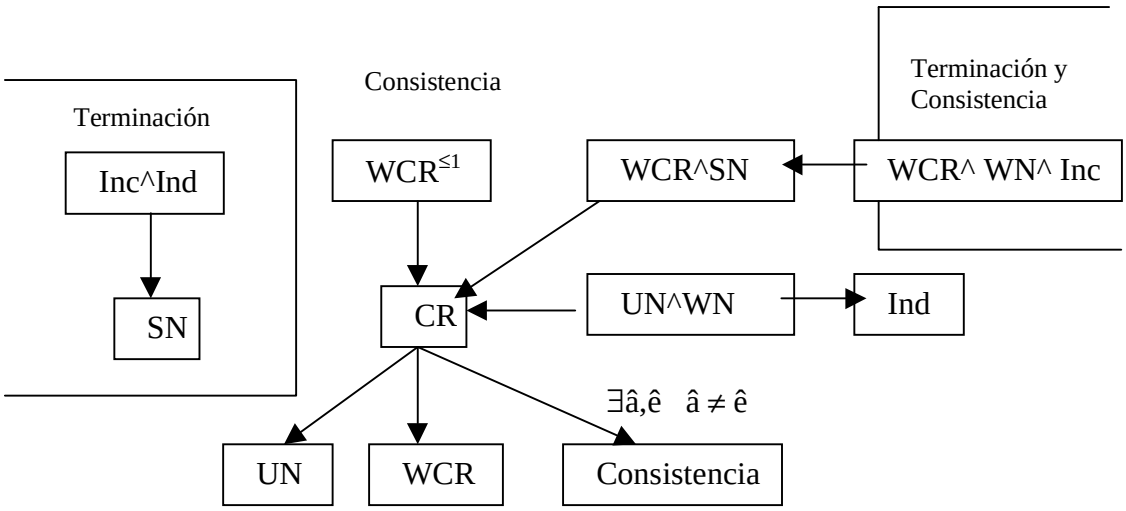
Inductividad Ind()

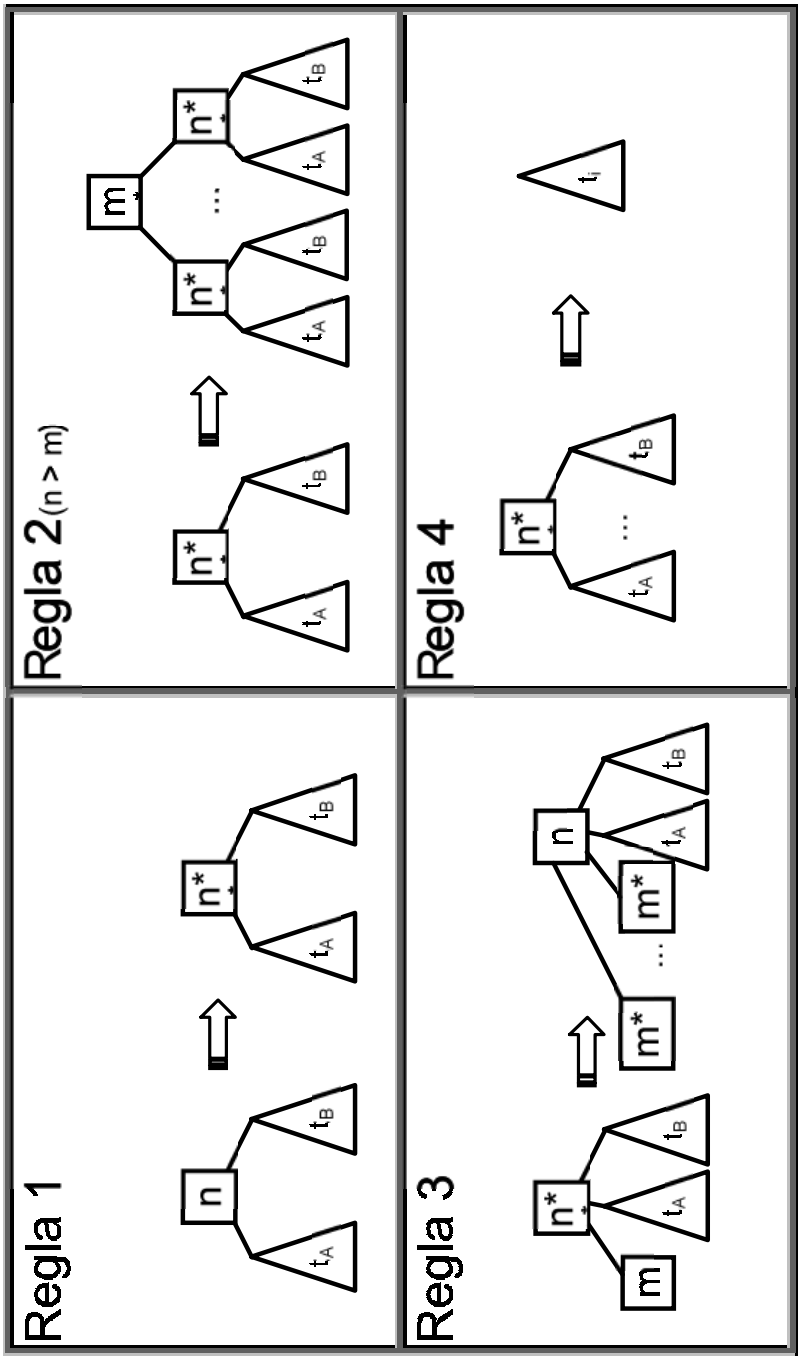
Sea una cadena infinita de la forma a →b →c →…, entonces se da que

a→ →ç, b→ →ç, c→ →ç, ……



GRAFO DE RELACIÓN DE PROPIEDADES




Regla 1
Regla 2(n > m)
Regla 3
Regla 4