UNIVERSIDADE DE A CORUÑA
FACULTADE DE INFORMATICA

TESE DOUTORAL

ALGORITMOS DE UNIFICACION ECUACIONAL EN TEORIAS XERAIS

Autor: LUIS CARLOS CACHAFEIRO CHAMOSA
Director: XOSE LUIS FREIRE NISTAL
A Coruña, Xullo 1994

FACULTADE DE INFORMATICA

TESE DOUTORAL

Algoritmos de Unificación Ecuacional en Teorías Xerais

Autor : Luis Carlos Cachafeiro Chamosa

presentada no
DEPARTAMENTO DE COMPUTACION

para a obtención do
GRADO DE DOCTOR EN INFORMATICA

Prof. Dr. D. Xosé Luis Freire Nistal

Tribunal Calificador

Presidente :	Prof. Dr. D. José Luis Maté Hernandez
Vocais :	Prof. Dr. D. José Alberto Jaén Gallego
	Prof. Dr. D. Miguel Angel López López
	Prof. Dr. D. Juan Pazos Sierra
Secretario :	Prof. Dr. D. Antonio Blanco Ferro

A Coruña, Xullo 1994

Os meus pais
A Pepi e a Alba

Agradezo os compañeiros do Laboratorio de Fundamentos da Computación e Intelixencia Artificial LFCIA do Departamento de Computación pola súa colaboración neste traballo. Agradezo en especial o profesor Freire polos seus consellos e adicación. Agradézolle tamén a Pepi polas "neuras" que tivo que soportar.

Indice

Capítulo 1

Introducción

1.1 Motivación

En computación, a unificación convertíuse nun campo de intensa actividade. Nunha forma abstracta, o proceso de unificación pódese expresar como :

> dadas as descripcións s y t achar os obxectos w que se adaptan ás descripcións s e t [107].

Trátase, polo tanto, dun mecanismo esencial para extraer información relevante a partires de pares de datos obrigados a ser identificados. Se a identificación non é posible, esta situación debe ser rápidamente detectada.

Se as instanciacións das expresións pódense facer coincidir sintácticamente, diremos que se trata da unificación de termos libres e se coinciden como elementos dunha mesma clase ecuacional nunha teoría E, entón se chamará E-unificación.

Na automatización do coñecemento soense empregar descripcións a xeito de reglas como :

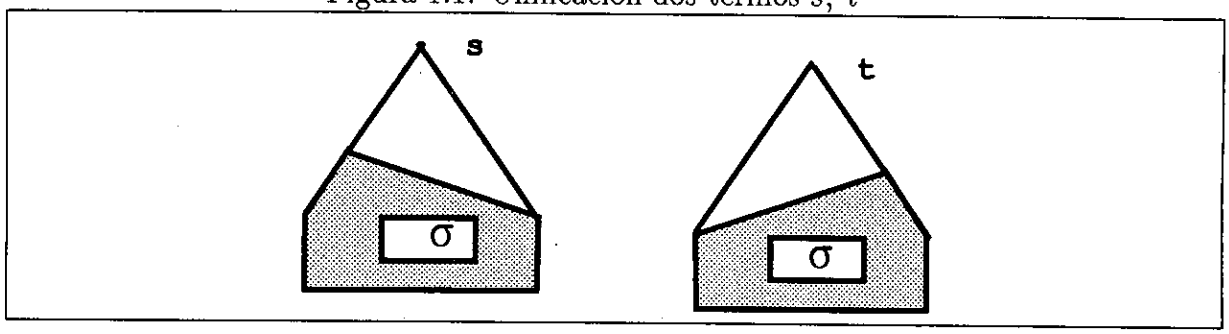
$$\begin{aligned} l_0 &\rightarrow r_0 \\ l_1 &\rightarrow r_1 \\ &\vdots \\ l_n &\rightarrow r_n \end{aligned}$$

para expresar a información acumulada nun determiñado campo.

Neste proceso de mecanización, soe ser fundamental coñecer se a descripción actual s pode facerse coincidir con algún dos l_i previos, en cuio caso, a regla $l_i \rightarrow r_i$ é activada. Noutros casos é importante saber se dúas descripcións dadas son representacións do mesmo obxecto.

Na figura 1.1, σ é a instanciación que identifica a s e a t e contén a información extraída, $(s)\sigma$ é a nova expresión resultado da unificación.

Figura 1.1: Unificación dos termos s, t



Esta operación de identificación úsase en campos como a visión por ordenador, o procesamento da lingoaxe natural, os sistemas de computación alxebraica, en deducción automática e outros [168, 107].

Nalgúns casos, a descripción actual é concreta (sen parámetros), mentras que en moitos outros os datos son flexibles ó admitir variables. A operación de identificación de obxectos no primeiro caso é coñecida como *matching*, e é polo tanto, un caso particular de unificación.

A potencia da unificación frente ás diversas formas de *matching* radica especialmente en que permite o paso das variables nos dous sensos. Ten na súa contra a propia complexidade do paso de información nas dúas direccións.

Cando as descripcións son a súa vez termos dunha álxebra ecuacional, a unificabilidade de s e t pódese ver como a xeralización dos métodos de resolución de ecuacións. Dada a gran cantidade de situacións posibles no caso ecuacional, baseadas nas propiedades das estructuras alxebraicas, desenroláronse diferentes técnicas para obter métodos de unificación ecuacional. En diferentes campos (proba automática, programación lóxica con igualdade, sistemas de reescrita módulo ecuacional entre outros) compren algoritmos de unificación ecuacional ben para teorías específicas ou ben para un grupo amplo de teorías. Noutros casos, en vez de utilizar directamente un método de unificación ecuacional, aproveitáronse mecanismos de ésta para o tratamento da igualdade neses campos (e.g. o cálculo **RPC** de Snyder e Lynch[172] na proba automática e o *narrowing* para enriquecer as lingoaxes funcionais con variables lóxicas e unificación[146, 128]).

Nesta tese relaciónanse os obxectos empregados na teoría de unificación cos da teoría de categorías. Moitos elementos da ciencia da computación nacen das investigacións nos campos da lógica e a álxebra. A teoría de categorías xurde da necesidade dos alxebristas e xeómetras por captar a razón intrínseca das súas propias construccións, basándose máis nas relacións entre os obxectos que nas súas propiedades internas. Por elo, convírtese nunha ferramenta eficaz para a descripción, pois permite englobar as diferentes lingoaxes matemáticas, sendo especialmente axeitado para as transformacións entre as distintas lingoaxes.

Na computación, a teoría de categorías é requerida pola xeralidade das súas construccións respecto da teoría de conxuntos, o que se observa no concepto computacional de función, mellor captado pola noción de morfismo nunha categoría, que a súa correspondente noción conxuntista. Amais, calquera acceso á estructura interna dos obxectos é expresamente evitada xa que as propiedades son consecuencia das relacións entre morfismos e non dos obxectos empreados. Na computación é fundamental a detección das propiedades xerais dun programa que son independentes da súa implementación, co obxectivo de obter unha semántica clara dos programas.

A teoría de categorías ofrece unha lingoaxe formalizada e axeitada para dar as propiedades abstractas das estructuras. Como sinala Rydeheard [159] :

> *Category theory is largely constructive. This means that category theory may be considered as computer programming, albeit of a rather high level of functionality. Readers of this should be aware that when they browse through textbooks on category theory they are reading very sophisticated programs!.*

Por elo, as categorías son usadas cada día con maior éxito na computación e.g. na especificación de programas; como ferramenta matemática na semántica das lingoaxes de programación; as categorías cartesiáns pechadas como semántica das lingoaxes tipadas de segundo orde[161, 5, 15].

1.2 Temas relacionados

A decidibilidade e o tipo de E-unificación dunha teoría dependen fortemente das súas propiedades alxebraicas. Un exemplo da complexidade da unificación ecuacional é a infinitariedade e decidibilidade da unificación asociativa[139], mentras que a unificación asociativa coa distributividade pola esquerda e pola dereita é indecidible[177] e a asociativa e conmutativa é finitaria.

Dada a complexidade da unificación ecuacional, teñen aparecido diversas técnicas para obter E-unificadores :

A primeira consiste no desenrolo de algoritmos específicos de E-unificación para teorías con propiedades dadas como a asociatividade, conmutatividade, idempotencia, identidade. Algunhas lingoaxes de programación como OBJ3 e MAUDE[76, 125], dispoñen de rutinas para o matching e a unificación en presencia destes axiomas, sendo tratados os axiomas como atributos das operacións.

Unha segunda técnica consiste na descomposición da teoría en subteorías, de forma que tódolos operadores e axiomas da primeira deben selo dunha soa das subteorías. Se existen algoritmos de unificación ecuacional para as subteorías, entón baixo certas condicións bastante xerais[11], pódense obter algoritmos de E-unificación para a teoría inicial. O problema xurde se se desexa introducir algún axioma que empregue operadores de varias subteorías, o que ten sido resolto recentemente, nun caso bastante simplificado, por Ringeissen[149]. Dada a complexidade da unificación ecuacional, cabe supoñer unha limitación na efectividade do método de combinación de teorías.

Unha tercera vía consiste en procurar a E-unificación no senso que Siekmann [168] denomina *universal unification algorithm* : Dada unha clase $\mathcal{E}$ de teorías, un algoritmo que acepta como entradas tanto ó par de termos a unificar como á teoría $E \in \mathcal{E}$ e xera un conxunto completo de E-unificadores do par. Incorporando ós axiomas específicos da teoría os axiomas de igualdade, a regra de resolución pódese ver como o primeiro algoritmo universal de E-unificación. O mecanismo de unificación universal máis coñecido é o de *narrowing*[64]. Este é completo para aquelas teorías cunha presentación canónica[90] e existe unha ampla variedade de técnicas que melloran o *narrowing* reducíndo o número de solucións redundantes[91, 147, 22] procurando a terminación[148, 37], posto que o procedemento de *narrowing* non é necesariamente finitario (que unha teoría sexa canónica non implica a finitariedade da súa E-unificación). A completitude do *narrowing* tense usado para tratar coas funcións nunha lingoaxe de programación lóxica con igualdade [76]. Outra forma de integrar a programación lógica e funcional, é a de incorporar a unha lingoaxe funcional un mecanismo de resolución, conservando a sintaxis funcional e usando narrowing para a semántica operacional[146, 128].

Kirchner[103] obtivo un método que proporciona algoritmos de E-unificación como extensión do algoritmo de unificación de termos libres de Martelli e Montanari[119]. Para elo, sobre un problema dado, actúan unha serie de regras de transformación, sendo necesarios tres tipos de regras para a unificación ecuacional : a descomposición do problema ou eliminación do operador superior, o mesturado ou reunión de termos unificados cunha mesma variable e as regras de mutación que empregan directamente os axiomas. Este terceiro paso é específico de cada teoría. Para as chamadas teorías sintácticas, Kirchner probou que a mutación pódese automatizar. As teorías sintácticas inclúen moitas das teorías coñecidas (como as asociativas e conmutativas) pero non é sequera semidecidible se unha teoría é sintáctica[106].

Para o caso xeral, tardaron en aparecer algoritmos universais de unificación ecuacional máis efectivos que a resolución cos axiomas de igualdade e que a paramodulación aplicada en calquera factor[38]. Algúns destes novos algoritmos, [71, 50] empregan unha variante do método de Kirchner, sustituíndo a mutación específica de cada teoría por unha nova regra de mutación moi laxa. Estes son chamados métodos *top-down* [71, 50] e pódese decir que as derivacións que se obteñen están guiadas polos *operadores superiores*. Os métodos *down-up* en cambio, imitan o mecanismo de narrowing, o cal é completo se se procede a unha complección sen fallo (*unfailing complection*) de pares críticos[72, 56, 129]. Os E-unificadores están dirixidos polos propios pares críticos e a secuencia que se obtén por estes métodos atópase bastante cerca da propia derivación que proba a igualdade das instanciacións.

Para realizar unha lingoaxe lóxica e funcional que combine os átomos ecuacionais e os predicados ordinarios, téñense empregado sistemas ecuacionais condicionais como nas lingoaxes BABEL e K-LEAF[128, 74], onde os predicados son considerados como funcións booleáns e o átomo p é equivalente á ecuación $p = true$. Nese caso, o procedemento de narrowing non é completo nen sequera no caso canónico se existen

variables extra (variables da parte condicional non presentes na cabeza). A unificación ecuacional en teorías xerais con presentacións condicionais, ata agora só ten sido estudiado por Delsart [50], cun método de E-unificación do tipo *top-down*.

Rydeheard e Burstall[161], estudiaron a unificación dende un punto de vista categórico, mostrando que o mecanismo de unificación de Robinson[150] é o resultado de aplicar, na categoría que chamaremos de substitucións [1], resultados xerais de teoría de categorías.

Baader[8] usou a categoría das álxebras de termos sobre conxuntos finitos e os seus homomorfismos para obter resultados acerca do tipo de unificación das teorías chamadas conmutativas. O seu complemento alxebraico víu dado por Nutt e Baader[12] onde se mostra a doble interpretación (alxebraica e categórica) destes resultados.

Os morfimos na categoría de substitucións, teñen moi restrinxido o seu dominio de actuación. Baader[9] denomina *unification restricted* a esta forma de unificación, na que os morfismos non poden mover calesquera variables, frente á *unification unrestricted* para a cal as sustitucións poden actuar tamén sobre conxuntos de variables fora do problema dado.

Rydeheard e Stell[162] mostraron que, empregando a noción de 2-categoría, o *narrowing* pódese incorporar á unificación ecuacional na categoría de substitucións. Na 2-categoría, as substitucións son 1-células e unha de elas pode ser remplazada por outra ecuacionalmente equivalente se existe algunha 2-célula que leve unha na outra. Sen embargo, dado o interés por empregar presentacións condicionais e que, como se probará, a noción de 2-categoría non é apropiada para incorporar a igualdade condicional, utilízase a noción de sesqui-categoría [?]. Stell considera que as sesqui-categorías son máis axeitadas para expresar a igualdade, o que se confirma ó probar aquí que, a diferencia da 2-categoría, a sesqui-categoría permite a incorporación das ecuacións condicionais como 2-células dela.

1.3 Obxectivo da tese

O principal obxectivo desta tese, é a obtención dun método de E-unificación na categoría de substitucións, que xeralice a forma de *narrowing* empregada por Rydeheard e Stell para teorías canónicas a teorías xerais. O novo algoritmo, pódese ver como :

- unha extensión da forma de narrowing empregada por Rydeheard e Stell, que permite amais o emprego de presentacións condicionais.

- un método de unificación ecuacional *down-up* máis depurado que os xa existentes, tanto por incorporar ecuacións condicionais, como pola eliminación de moitas das solucións redundantes dos anteriores métodos.

[1] aquí se diferenciará entre substitución e sustitucións.

Se se compara cos procedementos previos, obsérvase que mellora notablemente o realizado por Dougherty e Johann[56] que é o que impón condicións máis fortes á aplicación de axiomas. O que se desenrola nesta tese, atópase a medio camiño entre este e o propio narrowing. Para elo probaráse que : *se os problemas a resolver (tanto os iniciais como os intermedios) son lineais, o noso método coincide con narrowing*[2], mentras que *se os axiomas son lineais o de Dougherty e Johann e o aquí proposto son equivalentes.*

Dado que a linealidade dos axiomas pola esquerda ten sido imposta con algunha frecuencia (e.g. [87]) como unha das condicións relevantes para lograr a canonicidade, nos casos máis difíciles de tratar, é cando o noso método faise precisamente máis efectivo que o de Dougherty e Johann.

Método / Problema	Dougherty e Johann	Novo Método	Narrowing
linealidade unificandos e consecuencias	redundancias >	equivalentes < − − − >	
linealidade de axiomas	equivalentes < − − − >		non é completo
caso xeral	redundancias > completo	> redundancias completo	non é completo

Taboa 1. Comparación entre diversos procedementos de E-unficación

O maior control nas variables que se precisa ó traballar na categoría de substitucións, obliga a diferenciar entre as variables do problema dado e as dos axiomas empregados. Esta diferencia entre o papel dunhas e outras variables, é xustamente a que permite comparar e resaltar a forma de actuación do método proposto co de Dougherty e Johann e co narrowing.

Un breve resume deste traballo e sinalando as principais aportacións é a seguinte :

No capítulo 2 introdúcense as distintas estructuras empregadas, resaltando especialmente as propiedades das sustitucións relacionadas coa unificación. Unha variante da rescritura e da complección sen fallo, é utilizada para probar que o proceso de complección por pares críticos pode estenderse a presentacións condicionais e que proporciona relacións completas por niveles.

Introdúcense tamén algúns conceptos da teoría de categorías como as categorías de Kleisli e a categoría de substitucións. Dase unha caracterión dos morfismos nestas categorías. Para a igualdade de sustitucións esténdese á categoría de substitucións mediante a incorporación de morfismos (2-células) entre sustitucións (1-células). Próbase que a noción de 2-categoría non é apropiada para o uso da igualdade condicional e que, en cambio, a noción de sesqui-categoría sí soporta este xeito de presentación da igualdade.

[2]nese caso particular, o narrowing é pois tamén completo en teorías xerais.

No capítulo 3 estúdiase a unificación de termos libres e se recollen os principais algoritmos de unificación, mostrando diversas comparacións entre os mesmos. Estúdiase a forma de obter unificadores en forma *factorizada* que será empregada no novo método de unificación ecuacional. Introdúcese a unificación de pares de sustitucións e estúdianse as relacións entre as categorías de unificadores. Demóstrase que a propiedade dos unificadores máis xerais, de ser equivalentes ás sustitucións idempotentes, corresponde á existencia dunha adxunción entre as categorías de coigualadores e a dos coigualadores "idempotentes".

O capítulo 4 trata da unificación ecuacional, realizando en primeiro lugar un repaso das técnicas de E-unificación, resaltando o método de narrowing, as súas diferentes adaptacións e probando que é completo no caso de conxuntos pechados de presentacións condicionais. Estúdianse os distintos algoritmos universais para teorías xerais.

Desenrólase unha variante do mecanismo de abstracción (utilizado noutros procesos diferentes de unificación) que é empregada para realizar un método de unificación ecuacional semellante ó de Dougherty e Johann, aplicable á unificación de pares de substitucións. A partir da abstracción obtense un novo método de E-unificación que mellora considerablemente o de Dougherty e Johann e próbanse a corrección e completitude. Móstrase que o novo método permite amais a incorporación tanto de ecuacións condicionais como de novas melloras que non lle fan perder a completitude.

No capítulo 5, resúmense as conclusións deste traballo e suxírense novas líneas de investigación.

No Apéndice A, móstrase o código das funcións empregadas para unha implementación do novo algoritmo na lingoaxe de programación funcional CAML. Esto permite realizar unha comparación entre os mencionados algoritmos universais de E-unificación.

No Apéndice B, móstranse os resultados experimentais na resolución de diversos problemas prantexados, tanto no número de solucións redundantes, no número de nós xerados en cada derivación, como no tempo utilizado na resolución destes problemas.

Capítulo 2

Nocións e estructuras previas

Neste capítulo intróducense a maior parte das nocións básicas empregadas posteriormente. A razón de estudiar con certo detalle as sustitucións idempotentes é debido a que esa propiedade está fortemente relacionada cos unificadores máis xerais. Móstrase cómo conservar algunhas das propiedades da reescrita aínda nos casos máis problemáticos e próbase que as relacións que dirixen á igualdade condicional gozan da propiedade da completitude por niveles se se incorporan á igualdade os pares críticos xerados por un proceso *unfailing complection.* Considéranse algunhas definicións de teoría de categorías e dase unha caracterización dos morfismos en *Catsubst.* Para tratar coa igualdade ecuacional de substitucións, introdúcense as 2-categorías e as sesqui-categorías. Esta última permite incorporar as presentacións condicionais en condicións semellantes ás formadas exclusivamente por pares de termos.

2.1 Estructuras de datos

2.1.1 Conxuntos, Relacións e Arbores

Conxuntos e cadeas

Se X é un conxunto, $|X|$ expresará ó *cardinal*(X) e por $(x_1,\ldots,x_n)$ ou $< x_1,\ldots,x_n >$ á $n-upla\ ordenada$ de $x_1,\ldots,x_n$. Se X e Y son conxuntos, entón $X \times Y$ é o conxunto de tódolos pares ordenados (x,y) con $x \in X$ e $y \in Y$. Usaráse a notación $X \backslash Y = \{x \in X, x \notin Y\}$. O conxunto X é *unión disxunta* de Y e Z se estes conxuntos non teñen elementos en común.

Unha función $F : X \rightarrow Y$ é calquera subconxunto de $X \times Y$ tal que, para todo $x \in X$ existe exactamente un $y \in Y$ con $(x,y) \in F$, o *dominio* de F, $domF$ é X e o *rango* é $ran(F) = \{y \in Y | \exists x \in X, (x,y) \in F\}$. Para cada $x \in X$, o único $y \in Y$ tal que $(x,y) \in F$ denótase como $F(x)$, Fx ou ben $(x)F$, dependendo do contexto.

Para cada $B \subset Y, F^{-1}(B) = \{x \in X | Fx \in B\}$ e para cada $A \subset X, F(A) = \{Fx | x \in A\}, F/A = F \cap (A \times Y)$.

Se $y \in Y$, entón $F^{-1}(y) = F^{-1}(\{y\})$. Unha función $F : X \to Y$ é *inxectiva* se $(\forall x, x' \in X)(x \neq x' \Rightarrow Fx \neq Fx')$, *sobrexectiva* se $rango(F) = Y$ e *bixectiva* se é inxectiva e bixectiva. Se $F : X \to Y$ e $G : Y \to Z$ son funcións, entón

$$G \circ F = \{(x, z) | z = G(F(x))\}.$$

O conxunto $\{1, 2 \ldots\}$ de *enteiros positivos* denótase por $\mathbf{N}$. Se $k \geq 0$ é un enteiro e $F : X \to X$ entón $F^k = I$ (función identidade) se $k = 0$, e $F^k = F \circ F^{k-1}$ noutro caso.

Unha cadea sobre un conxunto X é unha aplicación de $\{1, 2, \ldots, n\} \to X$, X^* é o conxunto de todas as *cadeas* de X (*peche de Kleene*). Así, $\mathbf{N}^*$ é o conxunto de tódalas cadeas de enteiros positivos. A cadea vacía denotarase como $<>$. A cadea $< x_1, \ldots, x_n >$ expresa a función $F(i) = x_i$, $1 \leq i \leq n$.

A lonxitude dunha cadea w denótase como $|w|$. Para calquera cadea e calquera enteiro $k \geq 0$, defínese $k : w$ como $<>$ se $k = 0$; como as k primeiras entradas en w no mesmo orden que en w se $1 \leq k \leq |w|$; e como w se $k > |w|$. A k-ésima entrada da cadea ω será representada por ω_k.

A concatenación de cadeas indícase por un punto
$< x_1, \ldots, x_n > \cdot < y_1, \ldots, y_l > = < x_1, \ldots, x_n, y_1, \ldots, y_l >$. A cadea $< x_1, \ldots, x_n >$ tamén será representada por $x_1 \cdot \ldots \cdot x_n$. Emprégase a notación $\pi \cdot S$, $(S \cdot \pi)$ como abreviatura de $\{\pi \cdot \omega | \omega \in S\}$, $(\{\omega \cdot \pi | \omega \in S\}$ respectivamente).

Relacións

Unha relación $\to$ en X é un subconxunto de $X \times X$. Por $x \to y$ represéntase a $(x, y) \in \to$. Unha relación $\to$ en X ten a propiedade :

- *reflexiva* se $\forall x \in X(x \to x)$,
- *simétrica* se $\forall x, y \in X$ se $x \to y$ entón $y \to x$,
- *transitiva* se $\forall x, y, z \in X(x \to y$ e $y \to z$ entón $x \to z)$,
- *antirreflexiva* se $\forall x \in X(NON(x \to x))$,
- *total* se $\forall x, y \in X($ ou $x \to y$ ou $y \to x$ ou $x = y)$,
- *non ten cadeas infinitas descendentes (finitaria ou noetherián)* se non existe nengunha secuencia $x_1, x_2, \ldots$ de elementos de X tales que $x_i \to x_{i+1}$ para todo $i \geq 1$,
- *bó orden parcial*, abreviadamente *wpo* (*well-founded partial order*) se é transitiva e sen cadeas descendentes infinitas,
- un *bó orde* é un orden total coa propiedade *wpo*,
- *case-orden* se ten as propiedades reflexiva e transitiva.

Cando se teña un conxunto P de propiedades e unha relación dada, dirase que a relación $satisfaiP$ se satisfai cada elemento de P.

A $P-clausura$ dunha relación $\rightarrow$ en X, é a intersección de tódolos subconxuntos de $X \times X$ que conteñen dita relación e satisfán P. Denótase por :

- $\rightarrow^+$ á clausura transitiva de $\rightarrow$,
- $\rightarrow^*$ á clausura reflexiva e transitiva,
- $\leftrightarrow^*$ á clausura reflexiva, transitiva y simétrica.

Por simple inducción na lonxitude da secuencia, próbase que $x \rightarrow^+ y$ sempre e cando exista unha sucesión finita $x_1, \ldots, x_k, k \geq 2$, de elementos de X tales que $x_1 = x, x_k = y$, e se $1 \leq i < k$ entón $x_i \rightarrow x_{i+1}$.

Mediante o mesmo procedemento demóstrase que $x \rightarrow^* y$ sse[1] existe unha sucesión finita $x_1, \ldots, x_k, k \geq 1$, de elementos de X tales que $x_1 = x, x_k = y$, e se $1 \leq i < k$ entón $x_i \rightarrow x_{i+1}$.

A condición $x \leftrightarrow^* y$ verifícase sempre e cando exista unha secuencia finita $x_1, \ldots, x_k, k \geq 1$, de elementos de X tales que $x_1 = x, x_k = y$, y se $1 \leq i < k$ entón $x_i \rightarrow x_{i+1}$ ou $x_{i+1} \rightarrow x_i$. Cando sexa necesario destacar os elementos de tal secuencia indicarase (este último caso) como $x \leftrightarrow^* y(x_1, \ldots, x_k)$.

Sexa $\rightarrow$ unha relación en X, entón :

- $x \in X$ é *reducible* se $\exists y \in X (x \rightarrow y)$. En caso contrario dirase que é *irreducible*,
- x é unha *forma normal* para y sse $y \rightarrow^* x$ e x é irreducible. A forma normal do termo y, se é única, expresarase por $y\downarrow$,
- unha relación $\rightarrow$ en X é *confluente* se para todo $x, y \in X$, tal que $\exists z \rightarrow^* x$, $z \rightarrow^* y$ entón $\exists u \in X$, $x \rightarrow^* u$, $y \rightarrow^* u$,
- $\rightarrow$ en X é *Church-Rosser* se para todo $x, y \in X$ tal que $x \leftrightarrow^* y$ entón $\exists w, x \rightarrow^* w \ {}^*\!\leftarrow y$,
- $\rightarrow$ en X é *localmente confluente* se para todo $x, y \in X$, tal que se $\exists z \in X, x \leftarrow z \rightarrow y$ entón $\exists u \in X$, $x \rightarrow^* u$, $y \rightarrow^* u$,
- $\rightarrow$ é *canónica* se é finitaria e confluente.

2.1 ([87]) *Sexa X un conxunto e $\rightarrow$ unha relación en X tal que $\rightarrow^+$ é un wpo en X. Entón as seguintes condicións son equivalentes:*

a) para calquera $x_1, x_2, y \in X$, se x_1 y x_2 son formas normais de y entón $x_1 = x_2$,

b) para calquera $x, x_1, y, y_1, \in X$, se $x \rightarrow^* y$, x_1 é unha forma normal de x e y_1 é unha forma normal de y, entón $x_1 = y_1$,

[1] a expresión *sse* emprégase como abreviatura de *se e só se*.

c) $\rightarrow$ *é localmente confluente,*

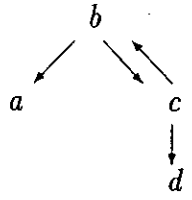
d) $\rightarrow$ *é confluente,*

e) $\rightarrow$ *verifica a propiedade de Chuch-Rosser.*

A equivalencia neste caso entre *c)* e *d)* é coñecida como *lema do Diamante*[2].

Se o orden non é un *wpo* non se pode conseguir a implicación *c)* $\Rightarrow$ *a)* como mostra o seguinte exemplo :

Exemplo 2.1 $X = \{a, b, c, d\}$



Arbores

As árbores son estructuras utilizadas na formación de termos e as súas propiedades aparecerán en distintas partes deste traballo. Un estudio detallado deste tipo de dato atópase nos traballos de Rosen e de Peterson e Stickel [155, 138].

Definición 2.1 *Sexan* $\pi, \omega \in \mathbf{N}^*$. *Defínense :*

- $Pa(\pi) = (|\pi| - 1) : \pi$ para $\pi \neq <>$ *(pai),*
- $Hiz(\pi) = Pa(\pi) \cdot < \pi_{|\pi|} - 1 >$ onde $\pi \neq <>; \pi_{|\pi|} \neq 1$, y $Pa(\pi) \cdot \pi_{|\pi|} = \pi$ *(irmán pola esquerda),*
- $\omega \preceq \pi$ *se* $|\omega| : \pi = \omega$ *(antepasado),*
- $\omega \prec \pi$ *se* $\omega \preceq \pi$ e $\pi \neq \omega$ *(antepasado estricto),*
- $\omega \perp \pi$ *se* $(\omega \not\preceq \pi$ ó $\pi \preceq \omega)$ *(independencia).*

Un conxunto $M \subset \mathbf{N}^*$ é *independente* se tódolos seus elementos son independentes entre sí.

Definición 2.2 *Un dominio de árbore é calquera subconxunto finito* D *de* $\mathbf{N}^*$ *que verifica que* $Pa(D) \subseteq D$ *e* $Hiz(D) \subseteq D$.

Definición 2.3 *Sexa* V *calquera conxunto. Unha árbore de membros de* V *é calquera función* $R : D \rightarrow V$ *na que* D *é un dominio de árbore. Os elementos de* D *son os nós da árbore.*

[2]este nome provén da demostración realizada por Huet.

O conxunto de tódolas árbores de membros de V denótase por V_* . Se R é unha árbore e π é un nó de $domR$, entón $R\pi$ representa o valor da árbore en π, ou sexa o "símbolo" no nó π.

Definición 2.4 Sexan $R,S \in V_*; \pi \in dom(R)$. Defínense

- $R/\pi = \{(\omega, x) | (\pi.\omega, x) \in R\}$ *(subárbore de R en π),*
- $R[\pi \leftarrow S] = \{(\omega, x) | (\omega, x) \in R \text{ e } \pi \not\preceq \omega\} \cup \{(\pi \cdot \gamma, x) | (\gamma, x) \in S\}$ *(reemplazamento en π de R por S).*

  Dado que R é a unión disxunta de $R_0 = \{(\omega, x) \in R, \pi \not\preceq \omega\}$ *y* $R_1 = \{(\omega, x) \in R | \pi \preceq \omega\}$, $R[\pi \leftarrow S] = R_0 \cup \{(\pi \cdot \gamma, x) | (\gamma, x) \in S\}$.

2.2 Sexan $\omega, \pi, \gamma \in \mathbf{N}^*$. *Entón pódese observar que :*

1. $\omega \preceq \pi$ *sse* $(\exists \gamma \in \mathbf{N}^*)(\omega \cdot \gamma = \pi)$;
2. *se* $\omega \preceq \pi$ *e* $\pi \preceq \gamma$, *entón* $\omega \preceq \gamma$ *(transitividade),*
3. *se* $\omega \preceq Pa(\pi)$, *entón* $\omega \prec \pi$,
4. *se* $\omega \preceq \pi$ *e tamén* $\omega \perp \gamma$, *entón* $\pi \perp \gamma$,
5. *se* $\pi \neq <>$, *entón* $Pa(\omega \cdot \pi) = \omega \cdot Pa(\pi)$.

2.3 *Sexan* $R, S, T \in V_*; \omega, \pi \in domR; \gamma \in domS$. *Verifícanse as seguintes propiedades :*

1. $R/\pi = S$ sse $R = R[\pi \leftarrow S]$,
2. $R[\pi \leftarrow S]/(\pi \cdot \gamma) = S/\gamma$ *(embebemento),*
3. $R[\pi \leftarrow S[\gamma \leftarrow T]] = R[\pi \leftarrow S][\pi \cdot \gamma \leftarrow T]$ *(asociatividade),*
4. *se* $\omega \perp \pi$ *entón* $R[\pi \leftarrow S]/\omega = R/\omega$ *(persistencia),*
5. *se* $\omega \perp \pi$ *entón* $R[\pi \leftarrow S][\omega \leftarrow T] = R[\omega \leftarrow T][\pi \leftarrow S]$ *(conmutatividade),*
6. $\gamma \in domR/\pi$ sse $\pi \cdot \gamma \in domR$,
7. *se* $\pi \in domR$ *e* $\gamma \in domR/\pi$, *entón* $(R/\pi)/\gamma = R/(\pi \cdot \gamma)$,
8. *para* $M \subseteq domR$ *un conxunto independente, defínese*

   $R[M \leftarrow S]$ *como R se* $M = \emptyset$, *noutro caso* $R[\omega_1 \leftarrow S] \ldots [\omega_n \leftarrow S]$ *onde* $(\omega_1, \ldots, \omega_n)$ *é calquera ordenación dos elementos de M,*

9. *calquera conxunto de follas en* $domR$ *é independente, onde* π *é por definición unha folla se* $domR \cap Pa^{-1}(\pi) = \emptyset$.

Proba :

1. Pola definición de $R[\pi \leftarrow S]$ este conxunto pódese expresar como a unión disxunta de R_0 e R_1 onde $R_0 = \{(\omega, x) \in R | \pi \not\preceq \omega\}$, $R_1 = \{(\pi \cdot \gamma, x) | (\gamma, x) \in S\}$. A árbore R é igual a $R_0 \cup R_1$ sempre e cando $R_1 = \{(\pi \cdot \gamma, x) \in R\}$. Pero esta condición corresponde xustamente a $R/\pi = S$.

2. Expresando novamente $R[\pi \leftarrow S] = R_0 \cup R_1$ de igual forma que en 1, $R[\pi \leftarrow S]/\pi \cdot \gamma = R_1/\pi \cdot \gamma = \{(\omega, x) | (\pi \cdot \gamma \cdot \omega, x) \in R_1\} = \{(\omega, x) | \gamma \cdot \omega \in S\} = S/\gamma$.

3. $R[\pi \leftarrow S[\gamma \leftarrow T]]$ pódese expresar como unión disxunta $R_0 \cup R_1^0 \cup R_1^1$, onde $R_0 = \{(\omega, x) \in R \ | \pi \not\preceq \omega\}$, $R_1^0 = \{(\pi \cdot \omega, x) | (\omega, x) \in S$ e $\gamma \not\preceq \omega\}$, $R_1^1 = \{(\pi \cdot \gamma \cdot \omega, x) | (\omega, x) \in T\}$ pero a mesma expresión representa a $R[\pi \leftarrow S][\pi \cdot \gamma \leftarrow T]$.

4. $R[\pi \leftarrow S] = R_0 \cup R_1$. xa que $\pi \not\preceq \omega$, $\omega \notin domR_1$. Dado que $\omega \not\preceq \pi, \omega \cdot \gamma \notin domR_1$, de forma que $R[\pi \leftarrow S]/\omega = R_0/\omega = R/\omega$.

5. $R[\pi \leftarrow S][\omega \leftarrow T] = R_0 \cup R_1$ onde $R_0 = \{(\gamma, x) \in R[\pi \leftarrow S] | \omega \not\preceq \gamma\}$ e $R_1 = \{(\omega \cdot \gamma, x) | (\gamma, x) \in T\}$. Expresando $R_0 = R_0^0 \cup R_0^1$, $R_0^0 = \{(\gamma, x) \in R |$ $\pi \not\preceq \gamma$, $\omega \not\preceq \gamma\}$, $R_0^1 = \{(\pi \cdot \gamma, x) | (\gamma, x) \in S$ e $\omega \not\preceq \gamma\}$ obtense que $R[\pi \leftarrow S][\omega \leftarrow T] = R_0^0 \cup R_0^1 \cup R_1$.

6. $\gamma \in domR/\pi$ sse $\exists x, (\pi \cdot \gamma, x) \in R$ sse $\pi \cdot \gamma \in domR$.

7. $(R/\pi)/\gamma = \{(\omega, x) | (\gamma \cdot \omega, x) \in R/\pi\} = \{(\omega, x) | (\pi \cdot \gamma \cdot \omega, x) \in R\} = R/(\pi \cdot \gamma)$.

8. Sexan $(\omega_1, \ldots, \omega_n)$, $(\omega_{\sigma(1)}, \ldots, \omega_{\sigma(n)})$ dúas ordenacións dos elementos de M (onde σ é unha unha permutación). Para $|M| = 1$ o resultado é naturalmente certo.

   Suposto para $|M|-1$. Se $\sigma(n) = n$ o resultado é certo directamente por inducción. Noutro caso :

   $$R[\omega_1 \leftarrow S] \ldots [\omega_{n-1} \leftarrow S][\omega_n \leftarrow S] = R[\omega_{\sigma(1)} \leftarrow S] \ldots [\omega_{\sigma(n)} \leftarrow S][\omega_n \leftarrow S]$$

   = aplicando a conmutatividade $(n - \sigma(n))$ veces $R[\omega_{\sigma(1)} \leftarrow S] \ldots [\omega_n \leftarrow S] \ldots [\omega_{\sigma(n)} \leftarrow S]$.

9. Sexan π, ω, $\pi \neq \omega$ dúas follas. Son independentes se $\pi \not\prec \omega, \omega \not\prec \pi$. Se $\pi \prec \omega$ entón $\omega = \pi.\gamma$ o que contradí a hipótese de que $domR \cap Pa^{-1}(\pi) = \emptyset$ □.

Definición 2.5 *Sexan* $\omega, \pi, \gamma \in \mathbf{N}^*$. *Se* $\pi = \omega \cdot \gamma$. *dise que* γ *é o cociente de* π *por* ω, *e expresarase por* $\pi/\omega = \gamma$

2.4 *Dados* $\omega, \pi \in \mathbf{N}^*, \omega \preceq \pi$, e $k \preceq \pi/\omega$. *Entón*

$\omega \cdot (\pi/\omega) = \pi$, $(\pi/\omega)/k = \pi/(\omega \cdot k)$.

Proba. Denotando por $\gamma = \pi/\omega, \pi = \omega \cdot \gamma$ directamente por definición de π/ω. Se $\gamma = k \cdot u$, tense $\pi = \omega \cdot \gamma = \omega \cdot k \cdot u$ e polo tanto $u = \pi/(\omega \cdot k)$.

2.5 *Sexan* $R,S,T \in V_*$; $\omega, \pi \in domR$; $\omega \preceq \pi$. *Entón*

1. $R/\pi = (R/\omega)/(\pi/\omega)$ *(cancelación),*
2. $R[\pi \leftarrow S]/\omega = (R/\omega)[\pi/\omega \leftarrow S]$ *(distributividade),*
3. $R[\pi \leftarrow S][\omega \leftarrow T] = R[\omega \leftarrow T]$ *(domiñancia).*

Proba.

1. Sexa $R/\pi = R/(\omega \cdot (\pi/\omega)) = (R/\omega)/(\pi/\omega)$ (aplicando *2.4* e o caso 7 de *2.3*).
2. Sexan $R_0 = \{(\omega \cdot \gamma, x) \in R[\pi \leftarrow S] | \pi \not\preceq \omega \cdot \gamma\}$, e $R_1 = \{(\omega \cdot \gamma, x) \in R[\pi \leftarrow S] \ | \exists k; \omega \cdot \gamma = \pi \cdot k; (k, x) \in S\}$. Sexan $R'_i = \{(\gamma, x) | (\omega \cdot \gamma, x) \in R_i\}$, $i = 0, 1$. Entón $R'_0 = \{(\gamma, x) | \pi/\omega \not\preceq \gamma\}$ e $R'_1 = \{(\gamma, x) = (\pi \cdot k, x) | (k, x) \in S\}$. Dado que $R[\pi \leftarrow S]/\omega = \{(\gamma, x) | (\omega \cdot \gamma, x) \in R_0 \cup R_1\}$ e xa que $\omega \preceq \pi$, $\gamma = (\pi/\omega) \cdot k$, tense $R'_1 = \{((\pi/\omega) \cdot k, x) | (k, x) \in S\}$. Pero desto dedúcese que $R'_0 \cup R'_1 = (R/\omega)/[(\pi/\omega) \leftarrow S]$.
3. Para probar a domiñancia, exprésase $R[\pi \leftarrow S][\omega \leftarrow T] = R_0 \cup R_1$, $R_0 = \{(\gamma, x) \in R[\pi \leftarrow S], \ \omega \not\preceq \gamma\} = \{(\gamma, x) \in R | \omega \not\preceq \gamma$ xa que $\pi \not\preceq \gamma\}$.
   $R_1 = \{(\omega \cdot k, x) | (k, x) \in T\}$. De onde $R[\omega \leftarrow T] = R_0 \cup R_1$ □.

Definición 2.6 *Sexa* $a \in V$. Defínese unha función $\bar{a} : (V_*)^* \rightarrow V_*$ *da forma seguinte: Sexa* $< R_1, \ldots, R_k > \in (V_*)^*$

$$\bar{a} < R_1, \ldots, R_k > = \{(<>, a)\} \left(\cup_{j \leq k} \{(j \cdot \omega, y) | (\omega, y) \in R_j\} \right)$$

Esta chamarase *árbore de raíz* a *e subárbores* $R_1, \ldots, R_n$. O caso $\bar{a}(<>)$ (abreviadamente $\bar{a}$) representa a árbore de raíz a e sen nengún outro nó.

2.6 *Dados* $a \in V$; $S, R_1, \ldots, R_n \in V_*$; $\omega \in domR_j$ *para algún* j *tal que* $1 \leq j \leq n$. *Entón*

$$\bar{a} < R_1, \ldots, R_n > [< j > \cdot \omega \leftarrow S] = \bar{a} < R_1, \ldots, R_j[\omega \leftarrow S] \ldots R_n > .$$

Proba. Pola definición de $\bar{a}$ para $i \neq j$ tense que $\bar{a} < R_1, \ldots, R_n > [j \cdot \omega \leftarrow S]/j \cdot \omega = S$ e $\bar{a} < R_1, \ldots, R_n > [j \cdot \omega \leftarrow S]/i = R_i$ con $(i \neq j)$. Pola primeira propiedade de *2.3* estas dúas igualdades caracterizan as dúas expresións que se desexa igualar □.

2.1.2 Termos e Sustitucións.

Introdúcense conceptos de termo e substitución así como algunhas definicións e propiedades derivadas. Unha referencia xeral é o traballo de Huet[87].

Pártese dun conxunto V de *símbolos de variables* e outro conxunto Σ de *símbolos de operadores* denominado *signatura*. Suporase que Σ é finito e que $V \cap \Sigma = \emptyset$, así como a existencia dunha función $ar : V \cup \Sigma \rightarrow \mathbf{N} \cup \{0\}$, tal que $ar(\nu) = 0$ para $\nu \in V$. Para calquera símbolo $x, ar(x)$ é a *aridade* de x.

Cando non exista confusión, denominarase *variable* e *operador* a un símbolo de variable e a un símbolo de operador respectivamente.

Definición 2.7 *Un termo é un membro do conxunto*

$$\mathcal{T}_\Sigma = \left\{ t \in (V \cup \Sigma)_* \mid \forall \pi \in dom(t) \mid dom(t) \cap Pa^{-1}(\pi) = ar(t\pi) \right\}$$

Danse as seguintes definicións e notacións :

- cando non sexa necesario destacar os operadores, o conxunto dos termos tamén se denotará por $\mathcal{T}$,
- t é un *termo concreto* se os seus símbolos son todos operadores i.e. $\forall \pi \in dom(t)$, $t\pi \in \Sigma$. o conxunto dos termos concretos escríbese como $\mathcal{T}_\Sigma(\emptyset)$ e é coñecido como *universo de Herbrand*,
- sexa $X \subseteq V$. O conxunto dos termos en X, $\mathcal{T}_\Sigma(X)$ (ou tamén $\mathcal{T}(X)$) é $\mathcal{T}_\Sigma \cap (X \cup \Sigma)_*$. O conxunto das variables de t, *Var(t)* está formado por todos os símbolos variables presentes en t, i.e. $Var(t) = \{v \in V | \exists \pi,\ t\pi = v\}$,
- identifícanse $t = \{(<>, s)\}$ e o símbolo s. De igual forma, non se diferenciarán t/π e $t\pi$ se π é unha folla de t. Asemade se $dom(t) \neq \{<>\}$, $t<> = f$, $ar(f) = k$, $t/<1> = t_1, \ldots, t/<n> = t_n$ entón denótase $t = f(t_1, \ldots, t_n)$,
- os operadores de aridade 0, serán chamados *constantes*. Nos operadores monádicos (de aridade = 1) omitiránse habitualmente os parénteses e rematarán nunha constante (iniciada por letras $a, b, c, d,$) ou nunha variable (letras x, y, z, w). E.g. $f(gx, a)$ é o termo $\{(<>, "f"), (1, "g"), (2, "a"), (11, "x")\}$,
- se t é un termo, o símbolo $t <>$ será chamado *operador superior de t*,
- un *subtermo* de t é unha subárbore de t que a súa vez tamén é un termo. Para indicar que s é subtermo de t utilizarase a notación $t[s]$. Para facer explícitas as variables $x_1, \ldots, x_n$ do termo t escribirase $t[x_1, \ldots, x_n]$, $Var(t) = \{x_1, \ldots, x_n\}$ e se s é outro termo arbitrario, $Var(t, s) = Var(t) \cup Var(s)$,
- t é *lineal* se $(\forall x \in V)(t/\pi = t/\omega = x \Rightarrow \pi = \omega)$,

- os elementos do dominio de t denomínanse as *posicións* de t. O dominio represéntase como $\Pi(t)$,
- unha posición π de t é *non variable* se $t\pi \in \Sigma$. O conxunto destas posicións exprésase por $\Pi_\Sigma(t)$,
- sexa s subtermo de t. Empregaranse as seguintes notacións para os conxuntos de posicións : $\Pi_s(t)$ (ou simplemente Π_s) é $\{\pi \in \Pi(t), t/\pi = s\}$, $\Pi_V(s) = \Pi(s) \backslash \Pi_\Sigma(s)$,
- sexa $\pi \in N^*$. Defínese recursivamente o *maior antecedente* ω *de* π *en* t por : $\omega = \pi$ se $\pi \in \Pi(t)$. Noutro caso $\omega =$ *maior antecedente*$(Pa(\pi))$,
- a *lonxitude* dun termo é a maior lonxitude das posicións dese termo, i.e. $lonx(t) = max\{|\gamma|,\ \gamma \in \Pi(t)\}$,
- o *tamaño* de t defínese polo número de operadores e variables (contando repeticións) que ten. Así, $tam(x) = 1$ se $x \in V$, $tam(f(t_1, \ldots t_n)) = 1 + \Sigma_{i=1}^{i=n} tam(t_i)$,
- o *tamaño estricto* de t é o número de operadores (incluíndo repeticións) que posúe. $tam\_estr(u) = 0$ se $u \in V$; $tam\_estr(f(t_1, \ldots t_n)) = 1 + \Sigma_{i=1}^{i=n} tam\_estr(t_i)$,
- a *extensión* de t, $ext(t) = tam(t) - |Var(t)|$.

Sustitucións

Definición 2.8 *Unha sustitución é unha aplicación* $\theta : \mathcal{T}_\Sigma \rightarrow \mathcal{T}_\Sigma$ *que verifica :*

- *se* c *é unha constante* $(c)\theta = c$,
- *se* $t \in \mathcal{T}$ $, t <> = f, ar(f) = n \neq 0, t = f(t_1, \ldots, t_n)$, *entón* $(t)\theta = f(t_1\theta, \ldots, t_n\theta)$,
- *existe un conxunto finito* $X = \{x_1, \ldots x_n\} \subset V$ *tal que* $\forall y \in V \backslash X,\ y\theta = y$.

Nesas condicións, dise que θ está *determinada por* X e escríbese $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$, onde $t_i = x_i\theta$, $1 \leq i \leq n$. Unha sustitución queda caracterizada unívocamente polos seus valores nas variables.

Para indicar a aplicación da sustitución θ ó termo t, usarase a forma $(t)\theta$, co fin de reservar a aplicación pola dereita á acción dun operador sobre seus argumentos.

Para simplificar a notación, escribirase tamén $t\theta$ en vez de $(t)\theta$. Se θ, ϕ son sustitucións, denotarase como $\theta \circ \phi$ (ou tamén $\theta\phi$), a sustitución composición : $(t)(\theta \circ \phi) = ((t)\theta)\phi$ e polo tanto, utilízase para a composición de sustitucións un orden inverso ó empregado para o das aplicacións. A mesma notación $\Gamma \circ \sigma$ e $\Gamma\sigma$, empregarase para compoñer a sustitución σ cun conxunto Γ de elas (pola dereita $\sigma \circ \Gamma$ ou $\sigma\Gamma$.)

Definición 2.9 *Se t, s son termos, e θ é unha sustitución, defínense :*

- $Dom(\theta) = \{v \in V | v\theta \neq v\}$ *(dominio de θ),*
- $I(\theta) = \{v \in V \,|\exists u \in Dom(\theta)\,, v \in Var(u\theta)\}$ *(imaxe de θ),*
- $Var(\theta) = Dom(\theta) \cup I(\theta)$ *(variables de θ),*
- $Rango(\theta) = \{v\theta | v \in Dom(\theta)\}$ *(rango de θ),*
- *sexa $W \subseteq V$, $W\sigma = \{x\sigma, x \in W\}$,*
- $\sigma \leq \theta$ *se existe λ, $\theta = \sigma\lambda$,*
- $\sigma \equiv \theta$ *se $\sigma \leq \theta$, e $\theta \leq \sigma$. Neste caso diráse que σ e θ son* equivalentes.

Debe notarse que θ está determinada por Dom(θ) polo que, $t\theta = t$ sse Var(t)$\cap$ Dom(θ) = $\emptyset$, e que Var($t\theta$) $\subseteq \big(Var(t) \setminus Dom(\theta)\big) \cup I(\theta)$.

Sexa $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ unha sustitución, t un termo e Var(t) $\cap$ $Dom(\theta) = \{x_1, \ldots, x_m\}$. Entón :

$$t\theta = t[\Pi_{x_1} \leftarrow t_1] \ldots [\Pi_{x_n} \leftarrow t_n].$$

Por elo, unha sustitución actuando sobre un termo só modifica o contido das posicións cuio símbolo é unha variable do dominio da sustitución.

2.7 *Sexa t un termo, θ unha sustitución e $\pi \in \Pi(t\theta)$. Entón ou ben $\pi \in \Pi(t)$ ou existen nós ω, γ tales que $\pi = \omega \cdot \gamma$, $\omega \in \Pi(t)$, $t\omega \in V$,* e *$\gamma \in \Pi((t\omega)\theta)$. No primeiro caso $t\theta/\pi = (t/\pi)\theta$. No segundo $t\theta/\pi = (t\omega)\theta/\gamma$ e dise que π está creada por θ.*

Proba. Se $\pi \in \Pi(t)$ o resultado obtense de aplicar inducción en $|\pi|$. Se $|\pi| = 0$ entón $t = t/\pi$ e $t\theta/\pi = (t/\pi)\theta$. Por outra parte, $t\theta/Pa(\pi) = (t/Pa(\pi))\theta$. Se $t/Pa(\pi) = f(t_1, \ldots, t_i, \ldots, t_n)$, tense que $t_i\theta = (t/\pi)\theta = (t/(Pa(\pi) \cdot i))\theta = (t/Pa(\pi))\theta/i = (t\theta/(Pa(\pi)))/\cdot i = (t\theta)/\pi$.

Noutro caso, sexa ω o maior antecedente de π en t. Polo tanto $\exists \omega, \gamma$; $\omega \in \Pi(t)$, $\omega \cdot \gamma = \pi$. Deséxase probar que $t\omega \in V$. Se $t\omega$ é constante $(t/\omega)\theta = t/\omega$ e por elo $\pi = \omega$. Se $t\omega \in \Sigma$ non é constante, entón $\omega \cdot i \in \Pi(t)$, $\omega \cdot i \prec \pi$ para algún i, o que contradí que ω sexa o maior antecedente de π en t. Sexa pois $v = t\omega$ unha variable. Pola propiedade anterior (para ω), $t\theta/\omega = (t/\omega)\theta = v\theta$. Así $t\theta/\pi = t\theta/(\omega \cdot \gamma) = (t\theta/\omega)/\gamma = (v\theta)/\gamma = ((t\omega)\theta)/\gamma$ □.

2.8 *Sexan s, t termos, $\pi \in \Pi(s)$, e θ unha sustitución. Entón :*

i) $\pi \in \Pi(s\theta)$,

ii) $(s/\pi)\theta = (s\theta)/\pi$,

iii) $(s[\pi \leftarrow t])\theta = (s\theta)[\pi \leftarrow t\theta]$,

iv) se para toda posición $\gamma \in \Pi(s), \gamma \perp \pi, s/\gamma \notin Dom(\theta)$, *entón* $s[\pi \leftarrow t]\theta = s[\pi \leftarrow t\theta] = s[\pi \leftarrow t\theta]$.

Proba. Inducción na lonxitude de $|\pi|$. Para $\pi = <>$, obviamente $\pi \in \Pi(s\theta)$, $(s\theta)/\pi = s\theta = (s/\pi)\theta$ e $(s[\pi \leftarrow t])\theta = t\theta = (s\theta)[\pi \leftarrow t\theta]$.

Suposto para $|\pi| - 1$, *i)* e *ii)* son semellantes ó caso anterior. Para *iii)* e da distributividade (*2.5*), dedúcese :

$(s[\pi \leftarrow t])\theta = \Big(s\big[Pa(\pi) \leftarrow \big(s/Pa(\pi)[i \leftarrow t]\big)\big]\Big)\theta = s\theta\big[Pa(\pi) \leftarrow \big(s/Pa(\pi)[i \leftarrow t]\big)\theta\big] = s\theta\big[Pa(\pi) \leftarrow \big(s\theta/Pa(\pi)[i \leftarrow t\theta]\big)\big] = s\theta[\pi \leftarrow t\theta]$ onde $i = \pi/Pa(\pi)$.

Para *iv)*, supónse que $s[\pi \leftarrow t]\theta \neq s\theta[\pi \leftarrow t\theta]$. Xa que $s[\pi \leftarrow t]\theta/\pi = t\theta = s[\pi \leftarrow t\theta]/\pi$ debe existir unha posición ω tal que $s[\pi \leftarrow t]\theta\omega \neq s[\pi \leftarrow t\theta]\omega$. As posicións $\omega \prec \pi$ verifican $(s[\pi \leftarrow t\theta])\omega = (s[\pi \leftarrow t\theta])\omega \in \Sigma$. Polo tanto, existe ω, $\omega \perp \pi, s[\pi \leftarrow t]\theta/\omega \neq s\theta[\pi \leftarrow t\theta]/\omega$. Se $\forall \omega \perp \pi, s\omega \notin Dom(\theta)$, entón $(s[\pi \leftarrow t]\theta)\omega = (s[\pi \leftarrow t\theta])\omega$ contradicción □.

2.9 *Sexan* θ, σ *sustitucións.*

1. *se* $I(\theta) \cap Dom(\sigma) = \emptyset, I(\sigma) \cap Dom(\theta) = \emptyset$ *e* $Dom(\theta) \cap Dom(\sigma) = \emptyset$, *entón* $\theta\sigma = \sigma\theta$,

2. *se* t *é un termo,* $I(\theta) \cap Dom(\sigma) = \emptyset$, *e* $Var(t) \subset Dom(\theta)$ *entón* $t\theta\sigma = t\theta$.

Proba. Se $x \notin Dom(\theta) \cup Dom(\sigma), (x)\theta\sigma = (x)\sigma = x = (x)\theta = (x)\sigma\theta$. Para $x \in Dom(\sigma)$ entón como $x \notin Dom(\theta)$, $x\theta\sigma = x\sigma = x\sigma\theta$, xa que $I(\sigma) \cap Dom(\theta) = \emptyset$. Análogamente para $x \in Dom(\theta)$.

Para 2. demóstrase primeiro que se $Dom(\theta) \cap Var(t) = \emptyset$ a sustitución θ é transparente para t, i.e. $t\theta = t$. En efecto, sexa π unha folla de t, que pode ser constante ou variable. Nos dous casos $(t/\pi)\theta = t/\pi$. Por *iv)* de *2.8*, $t\theta = t[\pi \leftarrow t/\pi]\theta = t[\pi \leftarrow (t/\pi)\theta] = t[\pi \leftarrow t/\pi] = t$. Xa que $Var(t) \subset Dom(\theta)$, entón $Var(t\theta) \subset I(\theta)$, e $Var(t\theta) \cap Dom(\sigma) = \emptyset, (t\theta)\sigma = t\theta$ □.

Tipos de sustitucións

Definición 2.10 *Se* $W \subset V$ *e* θ *é unha sustitución, a restricción de* θ *a* W *é a sustitución* θ_W *tal que para toda* $x \in V$:

- $x\theta_W = x\theta$ *se* $x \in W$,
- $x\theta_W = x$ *se* $x \notin W$.

Tamén se usará a forma $\sigma = \theta\ [W]$ *para expresar que* $\sigma_W = \theta_W$. *Para un conxunto de sustitucións* U, U_W *é o conxunto de sustitucións* $\{\sigma_W | \sigma \in U\}$.

Da definición dedúcese que $t\theta_W = t\theta$ se $Var(t) \subseteq W$.

Definición 2.11 *Sexan σ, θ, ρ sustitucións e $W \subset V$ dise que :*

- *θ é concreta se $I(\theta) = \emptyset$,*
- *σ é idempotente se $\sigma\sigma = \sigma$,*
- *ρ é un renomeamento se é inxectiva en $Dom(\rho)$ e $I(\rho) = Rango(\rho)$. Se $\rho = \{x_1/y_1, \ldots x_n/y_n\}$, $\rho^- = \{y_1/x_1, \ldots y_n/x_n\}$ é a sustitución inversa de ρ,*
- *unha permutación é un renomeamento bixectivo. Polo tanto, toda permutación ρ é un renomeamento que leva $Dom(\rho)$ en sí mesmo,*
- *a sustitución identidade representarase como ε,*
- *a unión $\sigma \cup \theta$ de σ e θ se $\sigma = \theta$ $[Dom(\sigma) \cap Dom(\theta)]$ defínese como : $x(\sigma \cup \theta) = x\sigma$ se $x \in Dom(\sigma)$, noutro caso $x(\sigma \cup \theta) = x\theta$,*
- *$\sigma \leq \theta[W]$ se existe λ, $\theta = \sigma\lambda\,[W]$.*

2.10 ([59]) *Calquera sustitución inxectiva que leve V en V é unha permutación.*

Proba. Se $x \in Dom(\sigma)$, entón $v = x\sigma \neq x$ con $v \in Dom(\sigma)$ pois $v\sigma = v = x\sigma$ con $v \neq x$ contradí a inxectividade. Como $Dom(\sigma)$ é finito, σ establece unha permutación de $Dom(\sigma)$ □.

2.11 *Unha sustitución σ é idempotente sse $Dom(\sigma) \cap I(\sigma) = \emptyset$.*

Proba. No caso de non ser idempotente $\exists x \in V, (x\sigma)\sigma \neq x\sigma$. Entón $\exists y \in Var(\sigma x), y\sigma \neq y$, de onde $y \in I(\sigma) \cap Dom(\sigma)$. Recíprocamente sexan $x, y \in Dom(\sigma), y \in Var(x\sigma)$. Entón $x\sigma\sigma \neq x\sigma$ e polo tanto σ non é idempotente □.

Se $\sigma = \{x_1/s_1, \ldots, x_n/s_n\}$ é idempotente, pódese expresar como composición de sustitucións simples (non importando a orde destas) $\sigma = \{x_1/s_1\}\{x_2/s_2\} \ldots \{x_n/s_n\}$. Outras caracterizacións das sustitucións idempotentes pódense atopar no traballo de Eder[59]. Nótese que a composición de sustitucións idempotentes non é necesariamente idempotente :

Exemplo 2.2 *$\sigma = \{x/f(y)\}$ $\tau = \{y/f(z)\}$, A composición $\tau\sigma = \{x/f(y), y/f(z)\}$ verifica $\tau\sigma\tau\sigma = \{x/f(f(z), t/f(z)\}$ e polo tanto non é idempotente.*

Non obstante, é posible, en certas condicións, asegurá-la idempotencia da composición:

2.12 *Sexan σ, τ sustitucións idempotentes con $Dom(\sigma) \cap I(\tau) = \emptyset$. Entón $\sigma\tau$ é idempotente.*

Proba. Sexa $x \in Dom(\sigma\tau)$. Se $x \in Dom(\sigma)$ tense que $x \notin I(\sigma), x \notin I(\tau)$ e así $x \notin I(\sigma\tau)$. Se $x \in Dom(\tau)$, tense $x \notin I(\sigma\tau)$ pola idempotencia de τ □.

Obsérvese que no exemplo anterior, $\sigma\tau$ é idempotente.

2.1.3 Instanciación, Matching e Unificación

Os termos pódense ordenar usando a instanciación : conversión dun termo noutro por unha sustitución. Os textos de Huet e de Eder[85, 87, 59] son referencias xerais deste apartado.

Definición 2.12 *Dados s, t termos, dise que t é unha* instanciación *de s e que σ é un matching de s* e *t se $s\sigma = t$. A existencia dun matching de s* e *t exprésase por $s \leq t$.*

2.13 *Sexa $s \leq t$. Se $s\sigma = t$, $s\theta = t$ entón $\sigma_{Var(s)} = \theta_{Var(s)}$.*

Proba. Sexa $x \in Var(s)$; $s\theta = t$. Sexa $\pi \in \Pi_x(s)$. Necesariamente $x\theta = t/\pi = x\sigma\Box$.

A relación $\leq$ é un case-orde, pois non verifica a propiedade simétrica nen a antisimétrica nen é, xeralmente total. Para esto último, obsérvase que $f(fx,y) \not\leq f(y,fx) \not\leq f(fx,y)$. Para a antisimétrica chega con coller dúas variables diferentes. Para a simétrica $x \leq f(x) \not\leq x$.

Definición 2.13 *Dados s,t termos e θ,σ sustitucións, e $W \subset V$, empregaranse as seguintes notacións :*

$s > t$ *se* $t \leq s$ *e* $s \not\leq t$,

$s \equiv t$ *se* $s \leq t$ *e* $t \leq s$,

$\sigma <_W \theta$ *se* $\sigma_W < \theta_W$,

$\sigma \equiv_W \theta$ *se* $\sigma_W \equiv \theta_W$.

Definición 2.14 *A sustitución θ, é :*

un unificador de s, t se $s\theta = t\theta$,

un unificador de $\Gamma = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$ se $s_i\theta = t_i\theta, 1 \leq i \leq n$. [3] *No que sigue, $Unif(\Gamma)$ expresará o conxunto dos unificadores de Γ,*

un unificador máis xeral ou mgu (most general unifier) de Γ se :

θ é unificador de Γ tal que :

se σ unifica a Γ, entón $\theta \leq \sigma$.

un unificador máis xeral de Γ restrinxido a W se é unificador de Γ e se σ unifica Γ entón $\theta \leq \sigma\,[W]$.

Pola transparencia das sustitucións sobre os termos que non posúen variables no seu dominio, tense que se s,t son termos tales que $Var(t) \cap Dom(\theta) = \emptyset$ e θ é un matching de s e t entón θ tamén é un unificador de s,t.

[3] o símbolo $\stackrel{?}{=}$ fai mención expresa de que se trata dun problema de unificación.

2.14 *Se* $s > t$ *entón* $ext(s) > ext(t)$

Proba. Probarase que se $s > t$ entón ou $tam(s) > tam(t)$ ou ben $tam(s) = tam(t)$ e $|Var(s)| < |Var(t)|$.

A diferencia entre s e t radica nos subtermos de t nos que se atopan as variables $x \in Var(t)$, pois se $\pi \in \Pi_\Sigma(t)$, entón $\pi \in \Pi_\Sigma(s)$. Así debe probarse que algunha variable que contribúe en $ext(t)$ fai medrar a extensión de s.

Se $|Var(t)| = 0$, non existe $s > t$. Sexa $x \in Var(t)$ e σ matching de t e s. Se $x\sigma$ non é variable, incrementa a extensión. Se polo contrario, as variables son levadas en variables, se $x\sigma = y$ entón se σ fose inxectiva, trátaríase dunha permutación e $t \equiv s$. Por elo, $\exists x, x' \in Var(t), x\sigma = x'\sigma$ e polo tanto, por inducción no número de variables, $|Var(s)| < |Var(t)|$ e $ext(s) > ext(t)$ □.

Como consecuencia :

2.15 ([87]) *A relación* $>$ *é noetherián.*

2.16 ([59]) *Se* $s \equiv t$ *existe unha permutación* ρ, $s\rho = t$.

Proba. Sexan λ, λ', $s\lambda = t$, $t\lambda' = s$ e $x \in Var(s)$. Necesariamente $x\lambda \in Var(t)$. Se $x\lambda = x'\lambda$ tense $x = x'$, xa que $x\lambda\lambda' = x = x' = x'\lambda\lambda'$. A permutación ρ é $= \lambda_{Var(s)}$ □.

Sexa Φ ([87]) unha bixección entre $\mathcal{T} \times \mathcal{T}$ e V. Defínese unha operación binaria $\wedge$ nos termos dada por:

$s \wedge t = f((t_1 \wedge s_1) \ldots, (t_n \wedge s_n))$ se $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$,

$s \wedge t = \Phi(s, t)$ noutro caso.

Se existe, a menor (resp. maior) das cotas superiores (resp. inferiores) dun conxunto, é o *supremo* (resp. ínfimo) do conxunto. Un *retículo* é un conxunto parcialmente ordenado no que todo par de elementos ten *supremo* e *ínfimo*. Un retículo é *completo* se todo subconxunto non vacío ten *supremo* e *ínfimo*.

Sexa $\hat{\mathcal{T}}$ o cociente $\mathcal{T}_\Sigma / \equiv$ completado cun maior elemento $\top$.

2.17 $[t \wedge s]$ *é o ínfimo de* $[s]$ *e* $[t]$.

Este ínfimo é denominado *xeralización más específica* en [139].

Proba. Sexan s, t termos, $s \not\leq t$, $t \not\leq s$. Por definición de Φ, $\Phi(s,t) \leq s$, $\Phi(s,t) \leq t$. Emprégase inducción na suma dos tamaños dos termos. Se $tam(s) + tam(t) = 2$ necesariamente $s = a$, $t = b$ con a e b un par de constantes diferentes. Sexa u unha cota inferior de s e t. Se $u < s$ entón $ext(u) = 0 < ext(s)$, u é variable e $u \equiv \Phi(s,t)$. Suposto para todo par de termos $tam(s') + tam(t') < tam(s) + tam(t)$. Se $t <>= f \neq g = s <>$, as cotas inferiores de s e de t son os $u \in V$ e polo tanto $u \equiv \Phi(s,t)$. Se $t = f(t_1, \ldots, t_n)$, $s = f(s_1, \ldots, s_n)$, as cotas inferiores son ou ben $u \in V$ ou $f(u_1, \ldots, u_n)$, con $u_1, \ldots u_n$ cotas inferiores de $(t_1, s_1), \ldots, (t_n, s_n)$ respectivamente. Aplicando inducción obtense o resultado requerido □.

2.18 *O conxunto de termos* $\hat{\mathcal{T}}$ *é un retículo completo.*

Proba. Sexa S un subconxunto non vacío de $\mathcal{T}_\Sigma(X)/\equiv$ e sexa s un elemento arbitrario de S. O número dos elementos t de $\mathcal{T}_\Sigma(X)/\equiv$ tales que $t < s$ é finito. Xa que cada dous elementos teñen *ínfimo*, obtense o ínfimo de todos eles facendo a intersección (que non é vacía xa que $[x] \leq [s]$, $\forall s \in S$).

Como se probará no seguinte capítulo, se dous termos teñen unha cota superior (distinta de $\top$) entón tamén teñen un *supremo*. Polo tanto, se $|S|$ é finito, o resultado obtense por inducción. Noutro caso, constrúese unha sucesión $S_1 \subset S_2 \subset \ldots \subset S_n \ldots, S = \bigcup_{i>0} S_i$ e $|S_i| = i$. Sexa $s_1 \leq \ldots \leq s_i \leq \ldots$ a sucesión de supremos. Se esta sucesión detense, este valor é o *supremo*. Noutro caso, dado que o número de elementos $\leq [t]$ de $\mathcal{T}_\Sigma(X)/\equiv$ é finito, a única cota superior é $\top$ e polo tanto este será o *supremo* □.

2.1.4 Equivalencia de sustitucións e idempotencia

As sustitucións idempotentes son especialmente axeitadas para representar unificadores. Este tipo de sustitucións e a súa relación coa unificación ten sido estudiada principalmente por Eder e por Lassez *et al.*[59, 113].

2.19 *Sexan* σ, τ *sustitucións tales que* $\sigma \equiv \tau$. *Existe unha permutación* ξ *tal que* $\sigma\xi = \tau$.

Proba. Sexan ρ, ρ' tales que $\tau = \sigma\rho, \sigma = \tau\rho'$. Os conxuntos $V\backslash Var(V\sigma), V\backslash Var(V\tau)$ son finitos, e pódese supoñer que $\exists\lambda : V\backslash Var(V\sigma) \rightarrow V\backslash Var(V\tau)$ inxectiva. Xa que $x\sigma\rho\rho' = x\sigma$ necesariamente $y\rho\rho' = y$, con $y \in Var(V\sigma)$. Como consecuencia $y\rho \in I(\sigma\rho) = I(\tau)$. Por elo $\rho_{Var(V\sigma)} : Var(V\sigma) \rightarrow Var(V\tau)$ leva y en $y\rho$ inxectivamente (posto que compoñendo con ρ' obtense y). A permutación ξ defínese como $\lambda \cup \rho_{|Var(V\sigma)}$ unión de dúas inxectivas, e polo tanto, ξ é unha permutación □.

2.20 *Para cada sustitución* σ *existe outra idempotente* τ, *tal que* $\sigma \leq \tau$.

Proba. Sexa z unha variable calquera non contida no dominio de σ. Defínese a sustitución $x\theta = z, \forall x \in Dom(\sigma)$. Pola elección de $z, Var(V\theta) \cap Dom(\sigma) = \emptyset$, e $Dom(\sigma) = Dom(\theta)$. Por elo, a composición $\sigma\theta$ ten o dominio incluido en $Dom(\sigma)$ e dado que $Var(V\sigma\theta) \subseteq Var(V\theta)$, necesariamente $Var(V\sigma\theta) \cap Dom(\sigma\theta) = \emptyset$, e polo tanto $\sigma\theta$ é idempotente □.

Dentro das sustitucións equivalentes, as idempotentes teñen un carácter minimal dado polo seguinte resultado :

2.21 *Se* σ, θ *son sustitucións idempotentes equivalentes* $\sigma \equiv \theta$, *entón :*

i) $|Dom(\sigma)| = |Dom(\theta)|$,

ii) $|I(\sigma)| = |I(\theta)|$,

Proba. Para *i)* se probará que existe unha aplicación inxectiva $h : Dom(\sigma) \rightarrow Dom(\theta)$. En efecto, sexa $x \in Dom(\sigma)$. Se $x \in Dom(\theta)$ defínese $h(x) = x$, noutro caso $x\sigma = z \in I(\sigma), x \neq z$, xa que $\exists \lambda, x\theta = x = (x\sigma)\lambda$, e defínese $h(x) = z$. Hai que probar que $z \in Dom(\theta)$, e o carácter inxectivo de h. Se $z\theta = z$, e tendo en conta que pola idempotencia $z \notin Dom(\sigma)$, tense que $x\theta = x = x\sigma\lambda = z\lambda$ e tamén $z\theta = z = z\sigma\lambda = z\lambda$, ou sexa $z = x$ o cal é unha contradicción.

Para probar o carácter inxectivo sexan $x, x' \in Dom(\sigma), x \neq x'$. Por construcción se $x, x' \in Dom(\theta)$ necesariamente $h(x) \neq h(x')$. Se $x, x' \notin Dom(\theta), x\theta = x = (h(x))\lambda$, $x'\theta = x' = (h(x'))\lambda$ e polo tanto $h(x) \neq h(x')$. Se $x \in Dom(\theta), x' \notin Dom(\theta)$, pola idempotencia de σ , $x \notin I(\sigma), h(x) = x \neq h(x') \in I(\sigma)$.

Para *ii)* utilízase tamén a construcción dunha aplicación inxectiva entre $I(\sigma) \rightarrow I(\theta)$. Sexa $x \in Dom(\sigma) \cap Dom(\theta), t[y_1, \ldots, y_m] = x\sigma$. Necesariamente $x\theta = t[z_1, \ldots, z_m]$ xa que $\exists \lambda, \lambda', \sigma\lambda = \theta$, $\sigma = \theta\lambda'$. Por elo $z_i = y_i\lambda$ deben ser m variables distintas. Fáiselle corresponder $h(y_i) = z_i$. Se $x \in Dom(\sigma), x \notin Dom(\theta)$, é semellante ó caso anterior con $h(z) = x$.
A proba do carácter inxectivo de h é semellante á do caso *i)* □.

2.22 *Sexa σ unha sustitución :*

1) se existe unha aplicación ρ de V en V tal que $(\rho(x))\sigma = x$ para todo $x \in Var(V\sigma)$ e $\rho(x) = x$ se $x \notin Var(V\sigma)$, entón ρ é unha sustitución e $\sigma\rho$ é unha sustitución idempotente equivalente a σ,

2) calquera sustitución idempotente que sexa equivalente a σ pódese expresar como $\sigma\rho$ onde ρ verifica a condición anterior.

Proba de *1)*. Xa que $|Dom(\sigma)|$ é finito, ρ é unha sustitución. Se $x \in Var(V\sigma)$ por definición $x\rho\sigma = x$. Polo tanto $\sigma\rho\sigma = \sigma$ de onde se obtén a idempotencia de $\sigma\rho$. Pola definición de ρ, ésta é un renomeamento en $Var(V\sigma)$. Por outra parte $\sigma\rho\rho^- = \sigma$.

Proba de *2)*. Sexa θ unha sustitución idempotente equivalente a σ e λ, λ' permutacións, tales que $\sigma\lambda = \theta$, $\theta\lambda' = \sigma$. Para definir ρ próbase que $\forall x \in Dom(\sigma) \cap I(\sigma)$ entón $\exists y \in Dom(\sigma)$, $y\sigma = x$. Se $x \in Dom(\theta)$ e polo tanto $x \notin I(\theta), x \in Dom(\lambda)$, $\exists y \in I(\theta) \cap Dom(\lambda'), y \notin Dom(\theta)$ e tal que $y\sigma = y\theta\lambda' = y\lambda' = x$.

Sexa $x \notin Dom(\theta)$. Se $x \notin I(\theta)$ é análogo ó caso anterior. Se $x \in I(\theta)$, e xa que $x\sigma \neq x, x\theta\lambda' = x\lambda' \neq x$. Posto que $x \in I(\sigma), x \in I(\lambda')$, existe $y \in I(\theta) \cap Dom(\lambda')$, $y\lambda' = x$ e $y\sigma = y\theta\lambda' = y\lambda' = x$. A sustitución ρ vén dada por $x\rho = y$ □.

Como consecuencia :

2.23 *Sexa σ sustitución non idempotente equivalente a unha idempotente θ. Entón $|Dom(\theta)| < |Dom(\sigma)|$.*

Proba. Sexa $y \in Dom(\sigma) \cap I(\sigma)$. Xa que a sustitución ρ empregada na propiedade anterior leva y en x para algún $x \in Dom(\sigma)$, $x\sigma = y$. Polo tanto, $x\sigma\rho = y\rho = x, x \notin Dom(\sigma\rho)$. Dado que $\forall x \notin Dom(\sigma)$ pódese definir $x\rho = x$, dedúcese que $Dom(\sigma\rho) \subset Dom(\sigma)$ e $|Dom(\sigma\rho)| < |Dom(\sigma)|$. O resultado séguese da propiedade *2.21* □.

A anterior proposición non se pode modificar para incluír a imaxe (contra do que cabe agardar de *2.21*). Como un exemplo de elo sexa $\sigma = \{y/f(z), x/y, u/y\}$, e $\theta = \{y/f(z), u/x\}$.

Observouse que as sustitucións idempotentes teñen un certo carácter minimal respecto do resto de sustitucións equivalentes. Unha caracterización das sustitucións equivalentes ás idempotentes obtense relacionándoas co *mgu* como se observa nos seguintes resultados:

2.24 *Se s, t son termos unificables entón existe un mgu σ idempotente e $Dom(\sigma) \subseteq Var(s, t)$.*

A demostración desta proposición verase no seguinte capítulo.

2.25 *Sexa θ un unificador máis xeral idempotente de Γ. Entón $\phi \in Unif(\Gamma)$ sse $\phi = \theta\phi$.*

Proba. Se θ é *mgu* idempotente de Γ, e $\phi \in Unif(\Gamma)$, existe λ tal que $\phi = \theta\lambda = \theta\theta\lambda = \theta\phi$. Se $\phi = \theta\phi$ entón $\theta \leq \phi$ e por conseguinte é unificador □.

Esta propiedade é a empregada por Robinson ([152]) para definir o unificador máis xeral.

2.26 *Se σ é unha sustitución, son equivalentes*

i) existe un conxunto de pares de termos Γ con σ como unificador máis xeral de Γ,

ii) existe unha sustitución idempotente que é equivalente a σ.

Proba.

i) $\Rightarrow$ *ii)*. E consecuencia directa de *2.24*.

ii) $\Rightarrow$ *i)*. Sexa $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ unha sustitución idempotente equivalente a σ. O conxunto de pares de termos $\{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$ ten como unificador a θ e por elo tamén a σ □.

2.1.5 Ecuacións e igualdade ecuacional

Definición 2.15 *Unha* ecuación *é un elemento de* $\mathcal{T} \times \mathcal{T}$. *A ecuación* (s,t) *escribirase* $s = t$. *Se* E *é un conxunto de ecuacións, entón* $E^{-1} = \{r = l \mid l = r \in E\}$. *Unha ecuación* $s = t$ *é* concreta *se* s *e* t *son termos concretos.*

Definición 2.16 *Se* E *é un conxunto de ecuacións, dous termos* s,t *verifican a relación* $s =^1_E t$ *se existe unha ecuación* $(l = r) \in E \cup E^{-1}$, *unha posición* $\pi \in \Pi(s)$, *e unha sustitución* θ *tal que*

$$s/\pi = l\theta \, , t = s[\pi \leftarrow (r)\theta].$$

Nestas condicións, denotarase tamén por $s =^1_E t \ (l = r, \pi, \theta)$ se é necesario indicar os detalles da ecuación, posición e sustitución. A relación $=_E$ defínese como a clausura reflexiva transitiva de $=^1_E$. Por ser $=^1_E$ simétrica, $=_E$ tamén o é e é polo tanto unha relación de equivalencia. A clase de equivalencia que contén ó termo t represéntase como $[t]_E$ ou tamén como $[t]$.

Dada a simetría da relación $=_E$, o conxunto de ecuacións E será interpretado como un conxunto de pares non orientados i.e. se $u = v \in E$ entón $v = u \in E$. No que segue, os conxuntos de ecuacións serán usados sen orientación.

2.27 *Sexan* $s,t \in \mathcal{T}$, θ *unha sustitución, e* E *un conxunto de ecuacións . Se* $s =_E t$ *entón* $s\theta =_E t\theta$ *(estabilidade).*

Proba. Por inducción na lonxitude k de $s = t_0 =^1_E t_1 =^1_E t_2 \ldots =^1_E t_k = t$. Para $k = 0 \, , s = t \, , s\theta = t\theta$ e $s\theta =_E t\theta$.

Suposto para $k-1$, entón $s\theta =_E t_{k-1}\theta \, , t_{k-1} =^1_E t(l = r, \pi, \phi) \, , t_{k-1}/\pi = l\phi, t = t_{k-1}[\pi \leftarrow r\phi]$. Pola propiedade *2.7*, $t_{k-1}\theta/\pi = (t_{k-1}/\pi)\theta = l\phi\theta \, , t\theta = \big(t_{k-1}[\phi \leftarrow r\phi]\big)\theta = t_{k-1}\theta[\phi \leftarrow l\phi\theta]$, de onde $t_{k-1}\theta =^1_E t\theta(l = r, \pi, \phi\theta)$ □.

2.28 *Sexan* $t \in \mathcal{T}, \{\pi_1, \ldots, \pi_n\}$ *un conxunto de posicións independentes do termo* t. *Se para todo* i, $1 \leq i \leq n \ t/\pi_i =_E s_i$, *entón* $t =_E t[\pi_1 \leftarrow s_1] \ldots [\pi_n \leftarrow s_n]$ *(monotonía).*

Proba. Primeiro se verá que se $t/\pi =_E s$ entón $t =_E t[\pi \leftarrow s]$. Para elo utilízase inducción na lonxitude da secuencia $t/\pi = t_0 =^1_E t_1 =^1_E \ldots =^1_E t_{k-1} =^1_E t_k = s$. Para $k = 0$ é consecuencia do caso 1 da propiedade 4.
Suposto para $k-1$, $t =_E t[\pi \leftarrow t_{k-1}] \, , t_{k-1} =^1_E s(l = r, \omega, \phi)$. Polo tanto $t_{k-1}/\omega = l\phi \, , t_k = t_{k-1}[\omega \leftarrow r\phi]$. Aplicando os casos 1 e 3 de *2.3* : $t =_E t[\pi \leftarrow t_{k-1}] = t[\pi \leftarrow t_{k-1}[\omega \leftarrow (t_{k-1}/\omega)]] =^1_E t[\pi \leftarrow t_{k-1}[\omega \leftarrow r\phi]](l = r, \pi \cdot \omega, \phi) = t[\pi \leftarrow s]$. De onde o lema se obtén de aplicar o resultado 4 de *2.3* e n veces o resultado anterior □.

Definición 2.17 *Sexan* s, t *termos e* E *un conxunto de ecuacións. Dise que* s *é unha* E-instanciación *de* t, *se existe unha sustitución* θ, *chamada* E-matching *de* s *e* t, *tal que* $s =_E t\theta$.

Dirase que s e t *son* E-unificables *se existe unha sustitución* θ *tal que* $s\theta =_E t\theta$ *e* θ *será chamado un* E-unificador *de* s e t. *A substitución* θ *é un E-unificador de* Γ *se é un E-unificador de cada par* $s \stackrel{?}{=} t \in \Gamma$. *Por* $U_E(\Gamma)$ *indícase o conxunto de tódolos E-unificadores de* Γ.

2.29 *Sexan* $s, t \in \mathcal{T}$, E *conxunto de ecuacións. Son equivalentes :*

i) s *é unha E-instanciación de* t,

ii) existe $s' \in [s]$, s' *instanciación de* t.

Se para un E dado e para todo termo t, $[t]$ é finito e existe un algoritmo que o xera, é posible determinar se un termo é ou non unha instanciación de outro (e polo tanto extraer o correspondente E-matching).

Cabe supoñer que nese senso, a E-unificación sexa semellante á E-instanciación e que, para obter os E-unificadores do par de termos s, t, sexa suficiente con buscar $s' \in [s]$, $t' \in [t]$ tal que s', t' sexan unificables. O problema non é tan simple (aínda para o caso das teorías para as que $[s]$ é finito e existe algún algoritmo que o xera). Como mostra delo :

Exemplo 2.3 $E = \{f(a) = g(a)\}$, $\Gamma = \{f(x) \stackrel{?}{=} g(x)\}$, *xa que* $[f(x)]_E = \{f(x)\}$, $[g(x)]_E = \{g(x)\}$.

Definición 2.18 *Sexan* θ, ϕ *sustitucións e* E *un conxunto de ecuacións.* θ *e* ϕ *son* iguales módulo E, *o que será expresado por* $\theta =_E \phi$, *se* $t\theta =_E t\phi$ *para todo termo* t. $\theta \leq_E \phi$ *se existe unha sustitución* λ, $\theta\lambda =_E \phi$. *Se* $\theta \leq_E \phi, \theta \neq_E \phi$ *represéntase por* $\theta <_E \phi$.

Sexa E un conxunto de axiomas na signatura Σ. Defínese $\mathcal{T}_{\Sigma,E}(X) = \mathcal{T}_\Sigma(X)/\equiv$. Os elementos deste conxunto non son necesariamente elementos de $\mathcal{T}_{\Sigma,E}$.

Exemplo 2.4 $\Sigma = \{f, g\}$ $X = \{x\}$, $E = \{f(y, z) = g(z)\}$. *O elemento* $\{g(x), f(x, x)\}$ *de* $\mathcal{T}_E(\{x\})$ *non coincide coa clase* $\{g(x), f(x, x), f(y, x), f(u, x) \ldots\}$ *en* $\mathcal{T}_E$.

2.1.6 Alxebras e Congruencias

Dada unha signatura Σ, unha Σ-*álxebra* A é un conxunto A e unha función $A^{ar(f)} \stackrel{f_A}{\to} A$ para toda $f \in \Sigma$. O conxunto A chámaselle *base* da Σ-*álxebra*.

Tanto $\mathcal{T}_\Sigma$ como $\mathcal{T}_\Sigma(X)$ son álxebras (de termos), onde $f_{\mathcal{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$, con $\mathcal{A} = \mathcal{T}_\Sigma$ no primeiro caso e $\mathcal{A} = \mathcal{T}_\Sigma(X)$no segundo.

Homomorfismos e Substitucións

Sexan A, B dúas álxebras de bases A, B respectivamente. A aplicación $\phi : A \to B$ é un Σ-*homomorfismo* (ou simplemente *homomorfismo*) se para todo $f \in \Sigma$ e calesquera $a_1, \ldots, a_n \in A$, $f_A(a_1, \ldots, a_n)\phi = f_B(a_1\phi, \ldots, a_n\phi)$.[4] Os *endomorfismos* son os homomorfismos nos que A = B. Os *isomorfismos* son aqueles homomorfismos que teñen inverso.

As sustitucións son endomorfismos na álxebra de termos nos que só varían un número finito de variables.

Dado que unha parte deste traballo, utiliza as sustitucións nunha forma máis restrinxida que a usual, ó ser necesario explicitar as variables do dominio e da imaxe, emprégase o nome de *substitución* para non confundir este tipo de homomorfismos, entre álxebras de termos sobre conxuntos finitos, coas sustitucións.

Definición 2.19 *Sexan X, Y conxuntos finitos de variables. Os homomorfismos de $\mathcal{T}_\Sigma(X)$ a $\mathcal{T}_\Sigma(Y)$ serán chamados* substitucións *de X a Y.*

O feito de facer explícitas tales variables é unha alternativa á dificultade que se crea nun problema (e.g. de E-unificación) polo feito de que unha sustitución pode mover calesquera variables, podendo ser usadas nunha regra ecuacional [95, 45]. En Common e Lescanne [45] empréganse sustitucións no senso habitual, así como T(X)-*substitucións* que corresponden ás mencionadas substitucións.

A relación entre ambas é moi forte como se mostra a continuación :

2.30 *Sexan X, Y conxuntos finitos de variables. Existe unha substitución σ de X a Y sse existe unha sustitución σ' tal que $Dom(\sigma') \subseteq X, I(\sigma') \subseteq Y$, e para todo $t \in \mathcal{T}_\Sigma(X)$, $t\sigma' = t\sigma$* e, *$X \setminus Dom(\sigma) \subseteq Y$.*

Proba. Dada σ esténdese a todo $\mathcal{T}$ a partir de $y\sigma' = y\sigma$ se $y \in X$ e $y\sigma' = y$ noutro caso. Dada σ' defínese $t\sigma = t\sigma'$, $t \in \mathcal{T}_\Sigma(X)$. □.

Definición 2.20 *Sexan Σ unha signatura, X un conxunto de variables e A unha Σ-álxebra de base A.*

unha Σ-asignación *é unha aplicación $X \xrightarrow{\psi} A$. Denotarase por $\eta_X : X \to \mathcal{T}_\Sigma(X)$ a asignación $\eta(x) = x$,*

unha Σ-álxebra inicial *A é unha Σ-álxebra, tal que dada calquera outra Σ-álxebra B existe un único Σ-homomorfismo $\phi : A \to B$,*

A é unha Σ-álxebra libre *sobre X se existe unha asignación $X \xrightarrow{\phi} A$, tal que para toda Σ-álxebra B e asignación $X \xrightarrow{\psi} B$, existe un único homomorfismo $A \xrightarrow{\lambda} B$, $\phi\lambda = \psi$.*

[4] os homomorfismos como as substitucións serán aplicados pola dereita.

2.31 *As Σ-álxebras iniciais, así como as Σ-álxebras libres sobre un X dado, son únicas agás isomorfismos.*

Proba. Se A, B son iniciais, entón existen homomorfismos únicos $A \xrightarrow{\psi} B, B \xrightarrow{\phi} A$. Pola unicidade $\psi\phi : A \to A$ debe ser a identidade de A e $\phi\psi : B \to B$ a identidade de B. Por elo $\phi = \psi^{-1}$.

Análogamente obtense o resultado para as libres □.

2.32 *Sexa $\psi : X \to A$ unha asignación arbitraria. Entón existe un único homomorfismo $\psi' : \mathcal{T}_\Sigma(X) \to A$ tal que $\eta \circ \psi' = \psi$.*

Proba. $(f(t_1, \ldots, t_n))\psi' = f(t_1\psi', \ldots, t_n\psi')$ está ben definida e é a única forma na que se garantiza que sexa homomorfismo □.

2.33 *A álxebra $\mathcal{T}_\Sigma(\emptyset)$ é a Σ-álxebra inicial e $\mathcal{T}_\Sigma(X)$ é a Σ-álxebra libre sobre X.*

Dada unha asignación $\phi : X \to A$, expresarase igualmente por ϕ o correspondente homomorfismo $T_\Sigma(X) \to A$.

Congruencias

Definición 2.21 *Sexa A unha Σ-álxebra. Unha relación binaria $\equiv$ en A é unha Σ-congruencia se se verifican :*

i) $\equiv$ é unha relación de equivalencia en A,

ii) dado $f \in \Sigma$, $ar(f) = k$, $a_i, b_i \in A$, $a_i \equiv b_i, i : 1 \ldots k$ Entón $f_A(a_1, \ldots, a_k) \equiv f_A(b_1, \ldots, b_k)$.

Unha congruencia $\equiv$ é completamente invariante *([33]) se dado calquera endomorfismo $\phi : A \to A$, tal que $a \equiv b$ entón $(a)\phi \equiv (b)\phi$.*

Exemplo 2.5 *$\Sigma = \{f\}$, $A = \{a, b, c\}$, $f_A(a) = a$, $f_A(b) = b$, $f_A(c) = c$. Defínese unha relación $R = \{(a, b)\}$. R xenera unha relación de equivalencia $\equiv$ que non é completamente invariante, xa que o endomorfismo $(a)\phi = c$, $(b)\phi = b, (c)\phi = c$, verifica que $a \equiv b, (a)\phi \not\equiv (b)\phi$.*

Toda Σ-congruencia sobre unha álxebra A, define unha Σ-álxebra cociente $A/\equiv$ dada por $f_{A/\equiv}([a_1], \ldots, [a_k]) = [f_A(a_1, \ldots, a_k)]$ onde $[a_i]$ representa á clase de a_i en $A/\equiv$.

Pola definición dada verifícase que $A/\equiv$ é unha Σ-álxebra e a aplicación canónica $A \to A/\equiv$, $a \rightsquigarrow [a]$ é un Σ-homomorfismo.

A conexión entre Σ-congruencias e Σ-homomorfismos, é de feito máis forte, pois dado un Σ-homomorfismo obtense unha Σ-congruencia.

Definición 2.22 *Sexa $\phi : A \to B$ un Σ-homomorfismo. O* núcleo *de ϕ defínese como a relación $a_1 \equiv_\phi a_2$ se $(a_1)\phi = (a_2)\phi$.*

2.34 *Sexa $\phi : A \to B$ un Σ-homomorfismo. Tense que :*

i) o núcleo de ϕ é unha Σ-congruencia,

ii) ϕ é un isomorfismo, sse ϕ é bixectiva,

iii) Se ϕ é sobre, entón $A/\equiv$ e B son isomorfos.

Proba :

i) sexan $a_1 \equiv_\phi b_1, \ldots, a_k \equiv_\phi b_k$ e f un operador de aridade k. Entón $(f_A(a_1, \ldots, a_k))\phi = f_B(a_1\phi, \ldots, a_k\phi) = f_B(b_1\phi, \ldots, b_k\phi)$ e polo tanto $f_A(a_1, \ldots, a_k) \equiv_\phi f_A(b_1, \ldots, b_k)$,

ii) a aplicación inversa de ϕ é tamén homomorfismo (e o homomorfismo inverso tamén é aplicación),

iii) a aplicación $A/\equiv_\phi \to B$ é claramente bixectiva e polo tanto, o homomorfismo correspondente é un isomorfismo □.

Sexa $\equiv$ unha congruencia en $\mathcal{T}$ completamente invariante. Tódolos endomorfismos en $\mathcal{T}/\equiv$ proceden de endomorfismos de $\mathcal{T}$.

2.35 *Sexa A unha Σ-álxebra e $\equiv$ unha Σ-congruencia completamente invariante. Entón :*

- *para todo endomorfismo ψ en A, a aplicación $\psi_\equiv : A/\equiv \to A/\equiv$, $([a] \rightsquigarrow [a\psi])$ é un endomorfismo. Se para todo $a \in A$, $a\psi \equiv a\varphi$ entón $\psi_\equiv = \varphi_\equiv$,*
- *Para calquera endomorfismo $\varphi : \mathcal{T}_\Sigma/\equiv \to \mathcal{T}_\Sigma/\equiv$ existe $\phi : \mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$ tal que $\varphi = \phi/\equiv$.*

Proba.

- $\psi_\equiv$ definida por $[a]\psi_\equiv = [a\psi]$ está ben definida por ser $\equiv$ completamente invariante e é endomorfismo como consecuencia de selo ψ,
- para erguer un endomorfismo en $\mathcal{T}_\Sigma$ dende φ constrúese primeiro a composición $\psi\varphi$ con $\psi : \mathcal{T} \to \mathcal{T}/\equiv$ o homomorfismo canónico. Para toda variable x, selecciónase un representante $t \in [x]\varphi$, $x\phi = t$. Por construcción de ϕ, verifícase a igualdade $\varphi = \phi/\equiv$ □.

Pódese observar que a relación $\equiv$ nos termos, dada pola *def. 2.13* non é realmente unha congruencia xa que $x \equiv y, y \equiv y$, pero $f(x, y) \not\equiv f(y, y)$.

2.1.7 Teorías ecuacionais

Definición 2.23 *Unha álxebra A* satisfai *a ecuación $s = t$, se para calquera asignación $\psi : X \to A$, $X = Var(s,t)$ se verifica que $s\psi = t\psi$. Unha* variedade ecuacional *de Σ-álxebras é unha clase de Σ-álxebras H tal que existe un conxunto de ecuacións E, para as cales H é a clase de álxebras que as satisfán. Tamén se dí que H é a* clase dos modelos *de E. A notación $A \models E$ expresa que A é un modelo do conxunto E. Unha ecuación $s = t$ é unha* consecuencia *do conxunto de ecuacións E, se se verifica en tódolos modelos de E. A* teoría ecuacional *$\mathcal{T}eo(E)$ defínese como o conxunto de tódalas consecuencias das ecuacións de E.*

Para indicar que $s = t$ é unha consecuencia do conxunto E de axiomas, emprégase tamén a notación $E \models s = t$ como abreviatura de que se $A \models E$, entón $A \models s = t$.

Dados dous conxuntos de ecuacións son equivalentes *se dan orixe á mesma teoría ecuacional.*

Birkoff[18] probou a existencia dunha álxebra libre xerada por X en toda variedade ecuacional.

Para obter as consecuencias dun conxunto de ecuacións E pódese empregar unha forma que fai explícita a reflexividade, transitividade, simetría, estabilidade e congruencia : Para elo defínese a deducción ecuacional $\vdash$ por :

i) $\vdash t = t$ para todo $t \in \mathcal{T}_\Sigma$ (axiomas reflexivos),

ii) $s = t \vdash t = s$ (axiomas simétricos), $s, t \in \mathcal{T}_\Sigma$,

iii) $s = t, t = u \vdash s = u$ (axiomas transitivos) $s, t, u \in \mathcal{T}_\Sigma$,

iv) Se f é operador k-ario, $(s_1 = t_1, \ldots, s_k = t_k) \vdash f(s_1, \ldots, s_k) = f(t_1, \ldots, t_k)$ (axiomas reflexivos funcionais),

v) Se σ é unha sustitución $s = t \vdash s\sigma = t\sigma$ (estabilidade).

Nótese que dado un conxunto de ecuacións E, a relación $\vdash$ coincide con $=_E$.

O teorema de Birkoff establece a equivalencia entre as ecuacións satisfeitas nos modelos da teoría, e as que se producen como consecuencia da relación $\vdash$ (completitude da deducción ecuacional).

2.36 *Sexa E un conxunto de axiomas. $E \models eq(s,t)$ sse $E \vdash eq(s,t)$.*

Unha *teoría ecuacional* pódese representar tanto por :

- as ecuacións dunha presentación,
- as ecuacións da teoría,
- a álxebra de termos $\mathcal{T}_E = \mathcal{T}_\Sigma / =_E$.

Se $E = \emptyset$, entón $\mathcal{T}_E = \mathcal{T}$ é a propia álxebra de termos e a igualdade ecuacional $=_E$ é a identidade sintáctica.

A medida que se incorporan ecuacións á teoría, esta faise máis restrinxida. De feito pode suceder que a álxebra de termos teña un único elemento en cuio caso dise que a teoría é *inconsistente*. Unha condición suficiente para a inconsistencia dunha teoría, é que incorpore axiomas do tipo $x = t, x \notin Var(t)$, xa que dados calesquera termos s, s' mediante dúas instanciacións diferentes σ, θ pódese facer $s =^1_E t(x = t, <>, \{x/s\})$, $s' =^1_E t(x = t, <>, \{x/s'\})$ e $s =_E s'$. Esta situación evítase impoñendo a condición de que as variables dos lados esquerdo e dereito dos axiomas, sexan as mesmas. Unha teoría cunha presentación desa forma dise *regular* (i.e. $Var(l) = Var(r)$ para toda $l = r \in E$). Unha teoría é *libre de colapso* se non contén ecuacións da forma $x = t, x \in Var(t), x \neq t$, (*ecuacións colapsadas*) o que protexe da inconsistencia producida por unha ecuación $x = t$, $x \notin Var(t)$. Moitas teorías consistentes, non son regulares nen libres de colapso. e.g. os axiomas $0 * x = 0, 1 * x = x$.

Unha teoría é *permutativa* se tódolos seus axiomas son *permutativos* i.e. $\forall l = r \in E$, e para toda $u \in \Sigma \cup V, |\Pi_u(l)| = |\Pi_u(r)|$ (cada símbolo aparece o mesmo número de veces en cada lado da ecuación). Unha teoría permutativa é *variable permutada* se tén unha presentación na que coinciden as posicións de cada operador e só varían as posicións das variables.

Unha ecuación da forma $f(x_1, \ldots, x_i, \ldots, x_n) = x_i$ coñécese como *proxección*. Unha teoría é *case libre de colapso* se para cada ecuación colapsada existe unha proxección co mesmo *operador superior*.

Unha teoría é *finita* se as clases de equivalencia da congruencia definida polas súas ecuacións, teñen cardinal finito. Unha teoría é *noetherián* se non existen cadeas infinitas descendentes de sustitucións polo orden $<_E$. Unha teoría é *Ω-libre* se para toda ecuación $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ necesariamente $s_i = t_i, i = 1, \ldots, n$. A teoría co axioma conmutativo é unha teoría finita e noetherián pero non Ω-libre.

Unha teoría é *simple* se non existen ecuacións que colapsen subtermos propios, $l = r \in E, \pi \in \Pi(l)$, se $\pi \neq <>$ entón $l/\pi = r$. Exemplo de teoría simple, é a asociativa.

Sexa E un conxunto de axiomas. E é unha *presentación sintáctica* (tamén chamada presentación resolvente) se para calquera secuencia $s_0 =^1_E s_1(l_0 = r_0, \pi_0, \rho_0), s_1 =^1_E s_2(l_1 = r_1, \pi_1, \rho_1), \ldots s_{n-1} =^1_E s_n(l_{n-1} = r_{n-1}, \pi_{n-1}, \rho_{n-1})$ como moito unha das posicións π_i é $<>$. Unha teoría é *sintáctica* se ten algunha presentación deste tipo. Estas foron definidas por Kirchner[103] quen mostrou a utilidade de dispoñer de presentacións sintácticas para simplificar a E-unificación neste tipo de teorías.

Unha presentación é m-sintáctica se para toda secuencia $s_0 =^1_E s_1(l_0 = r_0, \pi_0, \rho_0), s_1 =^1_E s_2(l_1 = r_1, \pi_1, \rho_1), \ldots s_{n-1} =^1_E s_n(l_{n-1} = r_{n-1}, \pi_{n-1}, \rho_{n-1})$ se verifica que $j = |\{\pi_i | \pi_i = <>\}| \leq m$. Unha teoría é m-sintáctica se dispón dunha presentación m-sintáctica.

2.2 Igualdade Ecuacional e Reescrita

2.2.1 Reescrita e Reescrita condicional

Reescrita

E coñecida a importancia da igualdade en campos como a proba automática, a computación simbólica, as lingoaxes de programación de alto nivel e outros. Sen embargo, o reemplazamento de iguais por iguais que da lugar á relación $=_E$ crea moitos problemas debido á simetría. Para limitar a explosión creada por esta e conservar o significado de tal predicado, utilízase a reescrita, que impón unha direccionalidade ó reemplazamento.

As relacións de reescrita, gozan de moitas das propiedades da igualdade e dende un punto de vista operacional corrresponden a métodos do tipo de encadeamento cara adiante. O seu mecanismo de inferencia deduce novos termos a partir de expresións coñecidas, a diferencia dos métodos que reducen os problemas a subproblemas e que requiren a miudo *backtracing*[84].

Unha referencia moderna e ampla acerca da reescrita ecuacional é o artigo de Dershowitz e Jouannaud[52].

Definición 2.24 *Unha relación* $\rightarrow$ *en* $\mathcal{T}$ *é unha* relación de reescrita *se é pechada para o reemplazamento e para a aplicación de sustitucións i.e. :*

- *se* $l \rightarrow r$ *e* $s \in \mathcal{T}, \pi \in \Pi(s)$ *tense que* $s[\pi \leftarrow l] \rightarrow s[\pi \leftarrow r]$ *(monotonía ou congruencia),*
- *sexan* $l \rightarrow r$ *e* σ *unha sustitución, entón* $l\sigma \rightarrow r\sigma$ *(estabilidade).*

A relación $=_E^1$ é evidentemente unha relación de reescrita, pero o que se pretende é atopar outras relacións $\rightarrow \subset =_E^1$ para as que $\leftrightarrow^*$ coincide con $=_E$.

Definición 2.25 *Sexa* R *un conxunto de pares de termos* $\{(l_1, r_1), \ldots, (l_n, r_n)\}$. *Dise que* s *se* reescribe *en* t, $s \rightarrow_R t$ *(tamén se respresenta por* $s \rightarrow_{\pi, l_i \rightarrow r_i, \sigma} t$*) se existe* $(l_i, r_i) \in R, \pi \in \Pi(s), \sigma$ *tal que :*

$$s/\pi = l_i\sigma,\ t = s[\pi \leftarrow r_i\sigma]$$

O subtermo s/π *(resp. a posición* π*) é un* termo redex *(resp.* posición redex*). Un subtermo* s' *de* s *é* innermost *(respectivamente* outermost*) se* s' *non ten subtermos redex propios (respectivamente non é subtermo propio de nengún subtermo redex de* s*). As posicións correspondentes a subtermos innermost e outermost son chamadas posicións* innermost e outermost *respectivamente.*

O conxunto R *denomínase* sistema de reescrita de termos *ou abreviadamente* sistema de reescrita. *A clausura transitiva de* $\rightarrow_R$ *é coñecida como* derivación de reescrita *ou simplemente* derivación.

Pódese ver que $\rightarrow_R$ é unha relación de reescrita, e que un termo non ten subtermos *redex* sse é irreducible pola relación $\rightarrow_R$.

Xeralmente inclúense nas relacións de reescrita as seguintes condicións para tódolos pares (l, r) de R :

1. $Var(r) \subseteq Var(l)$,
2. l non é variable.

Con elo obtéñense condicións necesarias para a terminación de $\rightarrow_R$, pois :

1. se $\exists x \in Var(r), x \notin Var(l)$ a relación $\rightarrow_R$ non é noetherián xa que non se impoñen condicións ás instanciacións de $Var(r) \backslash Var(l)$, podendo instanciarse no propio termo inicial. E.g. Se $x \in Var(r) \backslash Var(l), x\sigma = l$, tense $l \rightarrow_R r[l] \rightarrow_R r[r[l]] \rightarrow_R \ldots$,
2. se $l \in V$ entón toda posición dun termo é redex e polo tanto, a relación non é finitaria.

Para garantir a aplicabilidade da relación $\rightarrow_R$ é conveniente que a derivación en R sexa confluente e noetherián. Esta última requírese para lograr a computabilidade e a primeira para obter unha única forma normal de calquera termo. Un sistema de reescrita confluente e noetherián determina un procedemento de decisión para a correspondente teoría ecuacional.

O problema de atopar un sistema de reescrita canónico para unha teoría dada ten, en xeral, unha solución negativa, xa que existen teorías para as cales a igualdade non é decidible [118, 140]. Para a igualdade dos termos concretos (tamén chamado *problema das palabras*) a solución segue a ser negativa[120]. Incluso existen teorías cun procedemento de decisión para as cales non existen sistemas de reescrita canónicos. Un dos casos máis representativos é o daqueles que inclúen o axioma conmutativo, xa que nengunha orientación de este, permite obter a terminación.

Téñense proposto diversas alternativas para aproveitar as ventaxas da reescrita no caso de que non sexa posible, ou non se coñeza a forma de garantir a confluencia ou a terminación baixo nengunha orientación dos axiomas.

Definición 2.26 *Sexa R un sistema de reescrita e S un conxunto de axiomas. Dise que s se* reescribe en t módulo S, $s \rightarrow_{R/S} t$ *se existen* $(l, r) \in R, \pi \in \Pi(s), \sigma, s', t'$ *tal que :*

$$s =_S s';\ s'/\pi = l\sigma,\ t' = s'[\pi \leftarrow r\sigma],\ t =_S t'$$

O conxunto R chamarase sistema de reescrita módulo S.

E claro que $\rightarrow_R \subseteq \rightarrow_{R/S}$ e que $\leftrightarrow^*_{R/S}$ coincide con $=_{R \cup R^{-1} \cup S}$.

Xa que a reescrita módulo S utiliza a relación $=_S$, supónse a existencia dun procedemento de decisión para ela. Para garantir a terminación de $\rightarrow_{R/S}$ débense

impoñer certas condicións ós axiomas de S, como por exemplo non ter un axioma cun termo variable no lado dereito e varias posicións desa variable no outro ou ben ter axiomas regulares [52].

Téñense empregado distintas condicións sobre R e S para asegurá-la decibilidade de $=_{R \cup R^{-1} \cup S}$ (como en [112, 87, 138, 94]), e.g. se $=_S$ é decidible, entón $=_{R \cup R^{-1} \cup S}$, éo tamén se R/S é *Church-Rosser módulo S* onde ; *R/S* é *Church-Rosser módulo S* se $\leftrightarrow^*_{R/S} \subseteq \quad \rightarrow^*_{R/S} =_S \leftarrow^*_{R/S}$.

Dous casos específicos pero de gran importancia corresponden á reescrita asociativo-conmutativa e á reescrita canónica concreta. A primeira por :

- a propia importancia dos axiomas asociativo e conmutativo en tódolos campos de uso da reescrita e.g. para a representación de obxectos abstractos como os conxuntos e multiconxuntos,
- a propia dificultad de orientación,
- a existencia de algoritmos de unificación nesa teoría [176, 61].

A reescrita en presencia dos axiomas asociativo e conmutativo é o paradigma da separación entre a pura reescrita e a que se realiza módulo un conxunto de axiomas. Prácticamente todas os desenrolos desta última teñen sido ideados para aplicalos cos axiomas de asociatividade e conmutatividade. Como exemplo de elo son os traballos de Peterson e Stickel, Lankford e Ballantine, Huet, Steinbach, Kapur *et al.* [138, 112, 87, 173, 101].

A importancia da reescrita canónica concreta, débese ás ventaxas que posúe a álxebra inicial :

- disponibilidade dun orden total nos termos,
- posibilidade de orientar as instanciacións concretas de regras con variables que son, por sí mesmas, imposibles de orientar,
- posibilidade de reconstruir sustitucións a partir de instanciacións concretas, nalgúns casos,
- decibilidade da igualdade no caso de R finito e S vacío,
- simplicidade da álxebra de termos concretos.

Unha propiedade esencial da reescrita é a de ser finitaria. No caso de utilizar a reescrita módulo un conxunto de ecuacións, a terminación de R non implica necesariamente a de R/S. Para asegurá-la terminación pode usarse unha relación $>$ irreflexiva, pechada para a sustitución e o reemplazamento e que conteña á relación R. Para os casos máis simples a orientación das regras obtense dende un bó orden

nos termos usando a lonxitude, tamaño, ou extensión[5]. Pero existen axiomas que non poden ser orientados utilizando estas funcións nos termos e. g. o axioma $a = b$.

Xa Knuth e Bendix [108] utilizan unha relación compatible coa reescrita. A finalidade destas relacións é :

- indicar o senso no que deben de ser aplicados os axiomas,
- proporcionar condicións suficientes para determinar se unha relación de reescrita é finitaria,
- sustituir unha regra difícil de compatibilizar con órdenes coñecidos, por outra ou outras que poidan ser orientadas, ou ben incluila entre os axiomas.

Na actualidade existe un gran número de relacións empregadas para garantir a finitariedade en distintos casos: as baseadas en órdenes polinómicos e exponenciais, na extensión lexicográfica e de multiconxuntos, simplificación, case-órdenes, e subtermos [51, 156].

Reescrita condicional

Xa que un dos principais campos cara os que se dirixe a unificación ecuacional é a programación lóxica con igualdade, estúdiase a reescrita no caso de teorías presentadas por un conxunto de regras condicionais.

Definición 2.27 *Dados os termos* $t_1, \ldots, t_n, u_1, \ldots, u_n, l, r$, *a expresión :*

$$t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n \Rightarrow l \to r$$

chamarase regra condicional, *de cabeza* $l \to r$ *e de corpo* $t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n$. *Un conxunto* S *de regras condicionais, xera unha relación* $R_S = \cup_{0 \leq j} R_{S,j}$, *onde as relacións* $R_{S,j}$ *(tamén expresadas por* $\to_{R_{S,j}}$ *e* $\to_{R,S}$ *para* R_S*), veñen dadas por :*

- $R_{S,0} = \Big\{(s,t) |$ *existe* $t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n \Rightarrow l \to r \in S$, *unha sustitución* ρ *e unha posición* $\pi \in \Pi(s)$, *tal que* $s/\pi = l\rho$, $\forall i,\ 1 \leq i \leq n,\ t_i\rho = u_i\rho,\ t = s[\pi \leftarrow r\rho]\Big\}$,
- $R_{S,j} = R_{S,j-1} \cup \Big\{(s,t) |$ *existe* $t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n \Rightarrow l \to r \in S$, *unha sustitución* ρ *e unha posición* $\pi \in \Pi(s)$, *tal que* $s/\pi = l\rho$, $\forall i,\ 1 \leq i \leq n,\ t_i\rho \leftrightarrow^*_{R_{S,j-1}} u_i\rho,\ t = s[\pi \leftarrow r\rho]\Big\}$.

Se $s \to^*_{R,S} t$, *necesariamente* $s \to^*_{R_{S,j}} t$ *para algún* j. *O menor* j *que o cumple denomínase* profundidade *de* $s \to^*_{R,S} t$.

[5]para que o orden dado por estas sexa pechada para o reemplazamento, é preciso usar a reescrita na súa forma restrinxida i.e. l non variable e $Var(r) \subseteq Var(l)$ para toda regra $l \to r$.

Se non hai posibilidade de confusión acerca do conxunto de regras condicionais, o subíndice correspondente pode aparecer omitido.

E claro que tanto as $R_{S,j}$ como R_S son relacións de reescrita. Se non hai condicións en nengunha regra, $R_S = R_{S,j}, 0 \leq j$. Como no caso non condicional, a confluencia de R_S esixe que $Var(r) \subseteq Var(l)$.

Poden existir variables no corpo dunha regra condicional, non presentes na cabeza, sen por elo perder a confluencia.

Exemplo 2.6 *$S = \{a \to b,\ a \to c,\ f(x,b) \stackrel{?}{=} f(c,x) \Rightarrow b \to c\}$. R_S é confluente dado que $b \to_{R,S} c$.*

Tódalas variables dunha regra condicional, que non se atopen presentes en $Var(l)$ serán chamadas *variables extra.*

A relación de igualdade $=^1_E$ que se obtén a partir dunha relación de reescrita condicional é, como na reescrita sen condicións, $R \cup R^{-1}$. Se cU é un conxunto completo de E-unificadores de $\{t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n\}$, entón as relacións xeradas mediante a regra $t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n \Rightarrow l \to r$ poden ser obtidas, polo conxunto $\{l\theta \to r\theta, \theta \in cU\}$. Xa que existen teorías que non son finitarias[6], este conxunto pode dar lugar a unha presentación infinita de regras non condicionais.

Bockmayr[21] estende a reescrita condicional, á reescrita módulo ecuacións, construíndo a relación $\to_{(R/E)}$, en forma semellante á reescrita condicional xa sinalada.

No caso de existencia de variables extra, a canonicidade non é unha condición suficientemente forte como para obter resultados semellantes ós que se dan na unificación ecuacional nas teorías canónicas sen variables extra ou sen condicións.

Definición 2.28 *Sexa $R = \cup_{0 \leq i} R_i$ dado a partir dun conxunto de regras condicionais S. R é confluente por niveles se cada R_i é confluente. Se R é noetherián e confluente por niveles dise que é* completo por niveles.

Unha relación R pode ser confluente sen ser confluente por niveles. O exemplo *2.6* é unha proba de elo, xa que R é confluente e finitario, pero R_0 xerado por $\{a \to b,\ a \to c\}$ non é confluente. En cambio, se R é noetherián necesariamente cada R_i éo. A relación dada polas regras $x \stackrel{?}{=} f(x) \Rightarrow a \to b,\ c \to f(c)$, é confluente por niveles pero non é completa por niveles ó fallar a finitariedade.

Nótese que se R é confluente por niveles, dado un termo s poden existir diversas formas R_i normais para s. E.g. $a \to b,\ f(a) \stackrel{?}{=} f(b) \Rightarrow b \to c$, xa que a forma R_0-normal de a é b, mentras que a súa forma R_1-normal é c.

2.37 *Sexa R completo por niveles e $s\theta \leftrightarrow_{R_i} t\theta$. Existe unha derivación $s\theta' \stackrel{*}{\leftrightarrow}_{R_i} t\theta'$ onde para todo $x \in Var(s,t)$, $x\theta' = (x\theta)\downarrow_{R_i}$.*

[6] no senso de ter conxuntos finitos de E-unificadores minimais.

Pola confluencia, existe $u \in \mathcal{T}, s\theta \rightarrow_{R_i}^* u \leftarrow_{R_i}^* t\theta$.

Para cada $x\theta, x \in Var(s)$ non normalizado, se verifica $s\theta \rightarrow_{R_i}^* s\theta[\pi \leftarrow x\theta'], \pi \in \Pi_x(s)$. Pola confluencia e terminación dedúcese a existencia de $u' \in \mathcal{T}$, tal que para todo $x \in Var(s,t)$ $s\theta \rightarrow_{R_i}^* s\theta[\pi \leftarrow x\theta'] \rightarrow_{R_i}^* u' \leftarrow_{R_i}^* t\theta[\pi \leftarrow x\theta'] \leftarrow_{R_i}^* t\theta$ □.

2.2.2 Confluencia e conxuntos pechados

Un problema abordado inicialmente por Knuth e Bendix [108] é o de cómo modificar un sistema non confluente de regras para dispoñer dun sistema de reescrita confluente para a teoría que definen. Xeneralizacións do procedemento de Knuth e Bendix foron abordadas por Landford e Ballantine [112] en teorías co axioma conmutativo e teorías permutativas, así como por Huet [87] e por Peterson e Stickel[138]. Estes últimos probaron que, baixo certas condicións de compatibilidade, pódese aplicar o método de complección a un sistema de reescrita (R,S) se existe un algoritmo de E-unificación para a teoría xerada por S.

O proceso utilizado por Knuth e Bendix coñécese como *complección por pares críticos* e está amplamente referenciado e xeralizado en Buchberger[30].

Definición 2.29 *Sexan $(l,r),(l',r')$ variantes de regras $\in R$. R presenta o par crítico $(l\rho[\pi \leftarrow r'\rho], r\rho)$ se $\pi \in \Pi_\Sigma(l)$ e $\rho = mgu(l/\pi, l')$.*

Se $t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n \Rightarrow l \rightarrow r$, $t'_1 \stackrel{?}{=} u'_1, \ldots, t'_m \stackrel{?}{=} u'_m \Rightarrow l' \rightarrow r'$ son dúas regras condicionais e existe $\pi \in \Pi_\Sigma(l)$, ρ é un mgu de l/π e de l', diráse igualmente que R presenta o par crítico :
$t_1\rho \stackrel{?}{=} u_1\rho, \ldots, t_n\rho \stackrel{?}{=} u_n\rho$, $t'_1\rho \stackrel{?}{=} u'_1\rho, \ldots, t'_m\rho \stackrel{?}{=} u'_m\rho \Rightarrow (l\rho[\pi \leftarrow r'\rho], r\rho)$.

No caso de que para estas dúas reglas condicionales existe i, ρ, π, $1 \leq i \leq n, \pi \in \Pi_\Sigma(t_i)$ (resp. $\pi \in \Pi_\Sigma(u_i)$), $\rho = mgu(t_i/\pi, l')$ entón a expresión $t_1\rho \stackrel{?}{=} u_1\rho, \ldots, t_i\rho[\pi \leftarrow r'\rho] \stackrel{?}{=} u_i\rho, \ldots, t_i\rho \stackrel{?}{=} u_i\rho$, $t'_1\rho \stackrel{?}{=} u'_1\rho, \ldots, t'_m\rho \stackrel{?}{=} u'_m\rho \Rightarrow l\rho \rightarrow r\rho$ (resp. para $t_i\rho \stackrel{?}{=} u_i\rho[\pi \leftarrow r'\rho]$) tamén se dirá que é un par crítico de R.

2.38 ([108, 87]) *Sexa R é un sistema de reescrita e $s \leftarrow_R u \rightarrow_R t$, entón existe w con $s \rightarrow_R^* w \stackrel{*}{_R}\leftarrow t$ ou ben existe un par crítico $(\bar{l}, \bar{r})$, unha sustitución η e unha posición $\pi \in \Pi_\Sigma(u)$, tal que $s = u[\pi \leftarrow \bar{l}\lambda]$ e $t = u[\pi \leftarrow \bar{r}\lambda]$.*

Proba. Sexan $u \rightarrow s(\pi, l \rightarrow r, \rho)$, $u \rightarrow t(\pi', l' \rightarrow r', \rho')$. Dependendo da posición relativa de π, π' tense que :

1) se $\pi \perp \pi'$ entón $s \rightarrow^* s[\pi' \leftarrow r'\rho'] \leftarrow^* t$,

2) se $\pi' = \pi \cdot \omega$, $\omega \in \Pi_\sigma(l)$. Neste caso existe un par crítico $(l\delta[\omega \leftarrow r'\delta], r\delta)$, $\delta\lambda = \rho \cup \rho'$. Polo tanto $s = u[\pi \leftarrow r\delta\lambda] = u[\pi \leftarrow \bar{r}\lambda]$, $t = u[\pi \leftarrow r'\delta\lambda] = u[\pi \leftarrow l\rho][\pi' \leftarrow r'\rho'] = u[\pi \leftarrow (l\delta\lambda[\pi' \leftarrow r'\delta\lambda])] = u[\pi \leftarrow (l\delta[\pi' \leftarrow r'\delta])\lambda] = u[\pi \leftarrow \bar{l}\lambda]$.

3) $\pi' = \pi \cdot \omega \cdot \omega'$ tal que existe $x \in Var(l)$, $\omega \in \Pi_x(l)$. Neste caso, u redúcese primeiro aplicando $l' \to r'$ nas posicións $\pi \cdot \pi_x$, $\pi_x \in \Pi_x(l)$ e logo aplícase $l \to r$. Sexa ρ'' a sustitución dada por $x\rho'' = x\rho[\omega' \leftarrow r'\rho']$, $y\rho'' = y\rho, y \neq x$. Así $s \to^* u[\pi \cdot \omega \cdot \Pi_x(l) \leftarrow r'\rho'][\pi \leftarrow r\rho''] \leftarrow^* t$.

No caso condicional o resultado é igualmente certo e se obtén por unha proba similar. No caso 2) tódalas condicións $t_i\delta \stackrel{?}{=} u_i\delta$, $t'_k\delta \stackrel{?}{=} u'_k\delta$ do par crítico son verificadas por λ e polo tanto a derivación se verifica.

No caso 3), atendendo ás derivacións entre as condicións obsérvase que $t_i\rho'' \leftarrow^* r_i\rho \leftrightarrow^* u_i\rho \to^* u_i\rho''$ polo que $u[\pi \cdot \omega \cdot \Pi_x(l) \leftarrow r'\rho']$ se reduce usando $l'\rho'' \to r\rho''$ □.

Este importante resultado non só ten producido un procedemento para obter regras confluentes, senon tamén un sistema de derivación e proba : o razoamento por complección ou razoamento cara adiante. O procedemento de complección descríbese en [30] como :

Algoritmo 2.1 *Algoritmo de compleccion de pares criticos*
Entrada : Un sistema de reescrita R
Salida : Un sistema de reescrita confluente G

Sexa $G = R$
Sexa $B = \bigcup_{(u,v)\in G}\{$ *pares criticos de* $(u, v)\}$
Mentras $B \neq \emptyset$
  Facer {
    Sexa (s,t) = elemento de B
    Sexa $B = B \setminus \{(s, t)\}$
    Sexa $(s, t) = (s, t)\downarrow$ */* forma normal de (s,t) pola relacion G */*
    Se $s \neq t$ *enton*
      Facer {
        analizar (s, t)
        Sexa $B = B \bigcup_{(p,q)\in G}\{$*pares criticos de* (s, t) *e de* $(p, q)\}$
        Sexa $G = G \cup \{(s, t)\}$ } }

O paso de $analizar(s, t)$ é empregado tanto para orientar o par resultante, como para deter a execución do algoritmo se o par crítico non pode ser orientado preservando a terminación [108]. Huet [88] da unha proba completa da correción deste algoritmo cando se incorporan unha serie de regras de reducción para mellorar a eficacia.

Existen conxuntos de regras para os que o anterior procedemento non acaba. Este non é o único problema, xa que existen ecuacións, que son utilizadas para obter a igualdade de dos termos, para as que se precisa empregar a ecuación nos dous sentidos, dependendo este da instanciación resultante.

Para o caso dunha teoría dada por un conxunto finito de ecuacións concretas, e xa que existen ordenes totais nos termos concretos, toda ecuación é orientable. A

complección por pares críticos xera un procedemento de decisión para tales teorías (a decibilidade xa foi probada por Ackermann). O método de *complección sen fallo* (*unfailing completion*[13]) é unha maneira de aproveitar parcialmente esta situación, tamén no caso xeral, no que unha regra pode xerar conxuntos infinitos de pares críticos e.g. $f(g(f(x))) \rightarrow g(f(x))$. Este proceso de complección obtense a partir dun orden total nos operadores e variables e dunha forma para estender este orden total para tódolos termos, (e.g. o *recursive path ordering*[51]). Se se incorporan tódalas instanciacións dos pares críticos, obtense un procedemento de complección que non falla, xa que tódalas ecuacións xeradas son sempre orientables, se ben poden producirse infinitas ecuacións. Deste xeito, para toda a clase ecuacional dun termo, existe un elemento en forma normal e a igualdade é semidecidible. Este proceso de complección ten sido amplamente utilizado en diversos campos da unificación ecuacional (e.g.[72, 28, 25, 11, 149]).

Unha variante do proceso de complección sen fallo, é desenrolado por Dougherty e Johann[56] para construir relacións non simétricas e confluentes que permitan xerar a igualdade ecuacional. Aquí amplíanse os resultados de Dougherty e Johann, incorporando as ecuacións condicionais (regras condicionais que poden ser usadas nos dous sensos).

Definición 2.30 *Dada unha relación $\succ$ monótona, finitaria e confluente, dise que* dirixe *á relación C, se C é o peche simétrico de $\succ$. Neste caso C é* pechada.

Nótese que non se esixe na definición que $\succ$ sexa estable. Polo tanto, dita relación non será, en xeral, unha relación de reescrita. A restricción $Var(r) \subseteq Var(l)$ da reescrita, que impide orientar satisfactoriamente certas regras, non é neste caso necesaria. A cambio, a relación debe incluír tódalas instanciacións dunha regra (se ben diferentes instanciacións poden utilizarse en distintos sensos).

Exemplo 2.7 *A partir de $R = \{f(x, gx, z) = f(x, y, x)\}$, a ecuación é orientada na forma $l \rightarrow r$ na instanciación $f(a, ga, ga) \succ f(a, a, a)$ e $r \rightarrow l$ na instanciación $f(a, f(a, ga, a), a) \succ f(a, ga, a)$.*

Se $=_E^1$ é pechada, entón pode considerarse obtida a confluencia, se bén as instanciacións deberán orientarse atendendo á $\succ$.

Se $=_E^1$ é pechada e, se $s \succ t$, entón, $s =_E^1 t$ e polo tanto, existe unha ecuación $u = v$ en E, unha sustitución ρ e $\pi \in \Pi(s)$, tal que $s/\pi = u\rho$, $t = s[\pi \leftarrow v\rho]$, ou alternativamente, $t/\pi = v\rho$, $s = t[\pi \leftarrow u\rho]$ xa que $u = v$ pode ser utilizada nos dous sensos. Como para a relación $=_E^1$, o caso anterior poderá ser expresado por $s \succ t\,(u = v, \pi, \rho)$.

Definición 2.31 *Se $s \succ t\,(u = v, \pi, \rho)$, dirase que é* propia, *se para toda $x \in Var(u, v)$ $x\rho$ é irreducible.*

Propiedades dunha relación $\succ$ pechada que dirixe a $=_E^1$:

- se a teoría é consistente, toda variable é irreducible para $\succ$.
  Proba : Supoñendo que $x \succ t, x \in V$, debe existir unha ecuación $u = v \in E$ que, pola consistencia, $u \in V, u \in Var(v)$. De onde, $u\rho = x, x \in Var(t), x \succ t[x] \succ t[t[x]] \ldots$ [7] Contradicción.
  Elo mostra asemade que se $u = v \in E$, e $u\rho \succ v\rho$, entón u non é variable,

- pola confluencia e finitariedade de $\succ$ todo termo t é reducible a unha única forma normal $t\downarrow$,

- as reduccións propias teñen un senso minimal, análogo o das reduccións *innermost* [83].

2.39 *Sexan E un conxunto de ecuacións pechado, dirixido por $\succ$ e t un termo non irreducible. Entón, existe unha secuencia $t \succ^* t'$ propia.*

Proba. Xa que t non é irreducible, existe unha posición $\pi \in \Pi_\Sigma(t)$ que é reducible e cuios subtermos son todos eles irreducibles. Sexan polo tanto $u = v \in E$, ρ tal que $t/\pi = u\rho, u\rho > v\rho$ e $Y = Var(v) \backslash Var(u)$ e se define ρ' por $y\rho' = (y\rho)\downarrow$ se $y \in Y$, $x\rho' = x\rho$, $x \notin Y$. Sexa $t' = t[\pi \leftarrow v\rho']$. Xa que $t = t[\pi \leftarrow u\rho'], t =^1_E t'(u = v, \pi, \rho')$ tense que $t \succ t'$ é unha derivación propia, posto que $t' \succ t$ contradí a terminación de $\succ$ $\square$.

Sexa E un conxunto de ecuacións. Empregando o algoritmo de Knuth-Bendix, obviando o paso de *analizar* e tomando o par crítico sen orientación obtense a sucesión $E^0 \subseteq E^1 \subseteq \ldots \subseteq E^n \ldots$, onde $E^0 = E$, $E^i = E^{i-1} \cup Pares\text{-}críticos(E^{i-1}), i \geq 1$. O conxunto $\bigcup_{i \geq 0} E^i$ denotarase por $C(E)$. En forma análoga xérase o conxunto $C(E)$ de ecuacións condicionais dende o conxunto inicial E de ecuacións deste tipo. A partir de $C(E)$ constrúese, na forma indicada previamente, a relación $R_{C(E)}$ das igualdades (nun só paso) xeradas por regras condicionais de $C(E)$, tamén expresado como $=^1_{C(E)}$.

Sexa $l = r \in E$, $Var(r) \not\subseteq Var(l)$, o conxunto de pares críticos inclúe pares de variantes de r, xa que para todo x, y onde $x \in Var(r) \backslash Var(l)$ o par $(r, r[\Pi_x \leftarrow y])$ é un par crítico.

O seguinte resultado mostra a utilidade da definición da relación pechada.

2.40 *Para todo conxunto de ecuacións condicionais E, $R_{C(E)}$ é un conxunto pechado. Existe unha relación $\succ$ que dirixe $R_{C(E)}$ e que é completa por niveles.*

Proba. Defínese un orden total $>$ finitario en $\Sigma \cup V$ e esténdese a todo $\mathcal{T}$ a partir do orden lexicográfico en : *(tamaño, operador superior, orden lexicográfico nos argumentos).* Polo tanto : $s = f(s_1, \ldots, s_n) >_\mathcal{T} t = g(t_1, \ldots, t_m)$ se :

a) $tam(s) > tam(t)$, ou

[7] $t[x] \succ t[t[x]]$ dedúcese de aplicar a monotonía.

b) $tam(s) = tam(t)$ e $f > g$, ou

c) $tam(s) = tam(t)$, $f = g$ e $\exists i \in N, \forall j, 1 \leq j \leq i-1, s_j = t_j$, e $s_i >_T t_i$

Sexa $\succ = >_T \cap =^1_{C(E)}$. Debe probarse que $\succ$ é :

1) monótona,

2) finitaria,

3) confluente,

4) a súa clausura simétrica é $=^1_{C(E)}$.

1) Sexa $s \succ t$. Compre probar que $f(u_1 \ldots, u_{i-1}, s, \ldots, u_n) \succ f(u_1 \ldots, u_{i-1}, t, \ldots, u_n)$.

   - se $tam(s) > tam(t)$ entón $tam(f(u_1 \ldots, u_{i-1}, s, \ldots, u_n)) > tam(f(u_1 \ldots, u_{i-1}, t, \ldots, u_n))$,
   - se $tam(s) = tam(t)$, $s = g(s_1, \ldots, s_m), t = h(t_1, \ldots, h_o)$ e $g > h$ entón $tam(f(u_1 \ldots, u_{i-1}, s, u_n)) = tam(f(u_1 \ldots, s_{i-1}, t, u_n))$ e o resultado obtense polo caso c), xa que $\forall k \leq i-1, u_k = u_k$ e $s >_T t$,
   - o terceiro caso é semellante ó anterior.

2) O orden lexicográfico en *(tamaño,símbolos,léxico)* é finitario, e polo tanto $>_T$ éo. Xa que $\succ \subseteq >_T$, tense que $\succ$ tamén é noetherián.

3) Para chequear a confluencia, pola propiedade *2.1*, é suficiente comprobar a confluencia local.

   Sexan $s_1 \stackrel{?}{=} u_1, \ldots, s_n \stackrel{?}{=} u_n \Rightarrow l \to r$, $t_1 \stackrel{?}{=} u'_1, \ldots, t_m \stackrel{?}{=} u'_m \Rightarrow l' \to r'$ as regras condicionais empregadas nas derivacións $s \prec u \succ t$. Por elo, existen π, π', ρ, η onde

   $s_1\rho =_{C(E)} u_1\rho, \ldots, s_n\rho =_{C(E)} u_n\rho$, $t_1\eta =_{C(E)} u'_1\eta, \ldots, t_m\eta =_{C(E)} u'_m\eta$,
   $u/\pi = l\rho$, $u/\pi' = l'\eta$,
   $s = u[\pi \leftarrow r\rho]$, $t = u[\pi' \leftarrow r'\eta]$,
   $s_i\rho =_{C(E)} u_i\rho$, $t_j\eta =_{C(E)} u'_j$, $1 \leq i \leq n, 1 \leq j \leq m$.

   e onde as variables de ambas regras (tanto as da cabeza como as do corpo) teñen sido renomeadas. Dependendo das posicións relativas de π e de π' tense que :

   3.1) se $\pi \perp \pi'$ entón $s \succ u[\pi \leftarrow r(\rho \cup \eta)][\pi' \leftarrow r'(\rho \cup \eta)] \prec t$ pola monotonía de $\succ$ e a conmutatividade (*2.3*). Xa que se verifican as condicións $s_i\rho =_{C(E)} u_i\rho$ tamén se verifican $s_i(\rho \cup \eta) =_{C(E)} u_i(\rho \cup \eta)$ (e o mesmo para $t_j \stackrel{?}{=} u'_j$),

3.2) noutro caso pódese supoñer que $\pi \cdot \omega = \pi'$. Aquí ou ben $\omega \in \Pi_\Sigma(l)$ ou ω é unha posición variable ou creada por ρ.

3.2.1) no primeiro caso $(l/\omega)\rho = l'\eta$, polo que $\rho \cup \eta$ é unificador de $(l/\omega, l')$ e $C(E)$ presenta un par crítico. Aplicando a propiedade *2.38*, dado que os pares críticos están en $C(E)$, que o orden é total, e supoñendo sen pérdida de xeralidade que $l\rho[\omega \leftarrow r'\eta] \succ^* r\rho$ entón $t = u[\pi' \leftarrow r'\eta] = u[\pi \leftarrow l\rho[\omega \leftarrow r'\eta]] \succ^* u[\pi \leftarrow r\rho] = s$. Pola separación das variables, tamén se verifican as condicións $s_i(\rho \cup \eta) =_{C(E)} u_i(\rho \cup \eta)$ e as $t_i(\rho \cup \eta) =_{C(E)} u'_i(\rho \cup \eta)$,

3.2.2) no segundo caso $\exists x \in Var(l), l/\omega' = x, \omega' \cdot \varpi = \omega$. Polo tanto $l'\eta$ é un subtermo de $x\rho$. Xa que $l'\eta \succ r'\eta$, e gracias á monotonía, se $\rho' = \rho_{Var(l,r)\setminus\{x\}}$, $x\rho' = x\rho[\varpi \leftarrow r'\eta]$, $l\rho \succ l\rho'$, $u/\pi' \succ (u[\pi \leftarrow r\rho'])/\pi'$ necesariamente $u \succ^* s \succ u[\pi \leftarrow r\rho'], u \succ t \succ^* u[\pi \leftarrow r\rho']$. Xa que $\rho =_{C(E)} \rho'$, de $s_i\rho =_{C(E)} u_i\rho$ dedúcese que $s_i\rho' =_{C(E)} u_i\rho'$.

4) Polo carácter total de $>_T$ e pola definición de $\succ$ tense que $\overset{\succ}{\leftrightarrow}$ e $=^1_{C(E)}$ coinciden.

A relación $\succ$ é amais unha relación completa por niveles. Sexa $=^1_{C(E),i}$ a relación $R_{C(E),i}$ obtida na definición *2.28* para $S = C(E)$. Defínese $\succ_i = \ > \cap =^1_{C(E),i}$. Para probar a confluencia local de $\succ_i$, sustitúese na proba anterior a condición $s \prec u \succ t$ por $s \prec_i u \succ_i t$. Falla probar que as condicións empregadas nos pasos 3.1), 3.2.1), 3.2.2) son resoltas por algunha derivación de profundidade $< i$. En 3.1) e 3.2.1) as derivacións empregadas para resolver as condicións son as de $s_1\rho =_{C(E),i-1} u_1\rho, \ldots, s_n\rho =_{C(E),i-1} u_n\rho, \ldots, t_m\eta =_{C(E),i-1} u'_m\eta$.

O caso 3.2.2) próbase por inducción en i. Para simplificar a notación suporase que $n = 1 = m$.

As derivacións empregadas son $u \succ s \succ^* s[\pi \cdot \cdot\Pi_x(r) \cdot \varpi \leftarrow r'\eta] = u[\pi \leftarrow r\rho']$ e $u \succ t \succ^* t[\pi \cdot \Pi_x(l) \cdot \varpi \leftarrow r'\eta] \succ u[\pi \leftarrow r\rho']$. As únicas condicións a verificar que non se topaban nas derivacións previas (e xa que logo de profundidade $< i$) son as $s_1\rho' \overset{?}{=} u_1\rho'$, correspondentes á última derivación realizada. Por hipótese $s_1\rho = u_1\rho$. As únicas posicións para as que se pode producir unha desigualdade (i.e. $s_1\rho \neq u_1\rho$) son da forma $\omega_x \cdot \varpi$, $\omega_x \in \Pi_x(s_1)$ (ou ben de u_1). Para cada posición deste tipo pode suceder que :

- corresponda a unha instancia dunha variable de l, esto é existe $y \in Var(l) \cap Var(s_1, u_1)$, $\omega_y \cdot \nu = \omega_x \cdot \varpi$,
- corresponda a unha instancia dunha variable extra, de xeito que existe $y \in Var(s_1, u_1)\setminus Var(l)$, $\omega_y \leq \omega_x \cdot \varpi$,
- que sexa unha posición non variable de s_1 ou u_1.

Para as posicións que se encontran no primeiro caso, redúcense mediante $t_1 \overset{?}{=} u'_1, \ldots, t_m \overset{?}{=} u'_m \Rightarrow l' \rightarrow r'$ as posicións da forma $\pi \cdot \omega'_y \cdot \nu$, $\omega'_y \in \Pi_y(r)$ en s e as da forma

$\pi \cdot \omega_y \cdot \nu$ en t. Nótese que entón o novo matching para $l \to r$ é unha nova sustitución (que por simplicidade de notación segue a escribirse como ρ'), $y\rho' = y\rho[\nu \leftarrow r'\eta]$.

No segundo caso, as instancias da variable extra son modificadas por $y\rho' = y\rho[\nu \leftarrow r'\eta]$.

No terceiro caso, en vez de usar $s_1 \stackrel{?}{=} u_1, \ldots, s_n \stackrel{?}{=} u_n \Rightarrow l \to r$ emprégase o par crítico $s_1\delta \stackrel{?}{=} u_1\delta, \ldots, s_n\delta \stackrel{?}{=} u_n, t_1\delta \stackrel{?}{=} u'_1\delta, \ldots, t_m\delta \stackrel{?}{=} u'_m\delta \Rightarrow l\delta \to r\delta$. Dado que $\delta \leq \rho \cup \eta$, seguiráse empregando ρ' como a instanciación para as variables de l, r.

Se tras estas operacións, as condicións son resoltas sen precisar novas derivacións, a profundidade da derivación é agora 0. Noutro caso repítese o proceso ata lograr que os pares de expresións das condicións coincidan. Este proceso remata, dado que só son modificadas aquelas condicións ν de s_i, u_i para as que $s_i/\nu = l'\eta'$ (respectivamente $u_i/\nu = l'\eta$).

Suposto para $i-1$. Pola confluencia de $\succ_{i-1}$ pode supoñerse que u está en forma $i-1$ normal para $\succ_{i-1}$. Entón verifícase que $s_1\rho =_{C(E),i-1} u_1\rho$ e hai que topar unha derivación desa profundidade para $s_1\rho' \stackrel{?}{=} u_1\rho'$. Existe unha derivación de $s_i\rho \succ_i^* s_i\rho'$ cuias condicións son da forma $t_1\eta =_{C(E),i-1} u'_1\eta$ resoltas por unha derivación propia. En forma análoga ó caso $i = 0$ constrúese unha nova derivación $s_1\rho \succ_{i-1}^* s_1\rho'$. Deste xeito, obtense que $t \succ_i t[\pi \leftarrow r\rho']$ e polo tanto $\succ_i$ é confluente □.

Dado un termo t o cardinal do conxunto $\{s | t \succ s\}$ non é necesariamente finito.

Exemplo 2.8 $E = \{f(g(a)) = f(y)\}$, $t = f(g(a))$. *Entón* $t \succ f(z), \forall z \in V$.
O número de derivacións propias dende t sí debe ser finita, xa que polo orden imposto, existe unha menor variable no orden $>_T$, *dado que* $f(z), f(y)$ *forman un par crítico e* $f(z) \succ f(y)$ *se* $z > y$.

Pode observarse que, xa que o cardinal de Σ é finito, dado un termo t, o cardinal de $\{[s]_{\equiv} | t >_T s\}$ onde $\equiv = \leq \cap \geq$ é necesariamente finito. Elo proba que o número de derivacións propias $t \succ^* \ldots$ é igualmente finito.

2.3 Categoría de Substitucións e Igualdade

2.3.1 Categorías

Na teoría de categorías utilízanse extensamente probas de tipo constructivo e moitas veces doadas de algoritmizar. Por outra parte, as categorías proporcionan modelos especialmente interesantes de teorías alxebraicas e próximas a procesos de naturaleza finitaria e algorítmica. Permite realizar a integración de procesos aparentemente diferentes, polo que é apropiada para a xeralización e construcción de obxectos de tipo abstracto.

Unha referencia clásica de a teoría de categorías é a obra de Mac Lane [116]. Os textos de Barr e Wells e de Asperti e Longo[15, 5] son referencias recentes da relación das categorías coa computación.

Definición 2.32 *Unha* categoría C *é un par de clases* $Ob(C)$ *e* $Mor(C)$ *de obxectos e morfismos de* C *respectivamente, tales que :*

- *para todo* $f \in Mor(C)$ *existen dous obxectos* dominio e codominio *de* f, *determinados en forma única por* f. *Se o dominio de* f *é* a *e o seu codominio é* b, *escribirase* $f : a \to b$, $dom(f) = a$, $cod(f) = b$,

- *para cada par de obxectos* (a, b), *existe un conxunto* $C[a, b]$, *dos morfismos de dominio* a *e codominio* b,

- *sexan* $f : a \to b, g : b \to c$ *morfismos calesquera, existe un único morfismo* $g \circ f : a \to c$ *que se chamará composición de* f *con* g. *A composición debe verificar o seguinte axioma:*
  $\forall f, g, h \in Mor(C)\ f \circ (g \circ h) = (f \circ g) \circ h$, *se* f, g, h *son componíbeis,*

- *para todo obxecto* a *de* C *existe un único morfismo denominado identidade de* a, 1_a *que verifica:*

  Para todo morfismo $f : a \to b$, *cúmplese* $1_b \circ f = f \circ 1_a = f$.

Definición 2.33 *Sexa* C *unha categoría e* $\circ$ *a composición en* C.

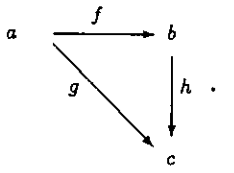
- *o morfismo* $f : a \to b$ *é* épico *se* $\forall g, h : b \to c$ *tal que* $g \circ f = h \circ f$ *entón* $g = h$,
- *o morfismo* $f : a \to b$ *é* mónico *si* $\forall g, h : c \to a$ *con* $f \circ g = f \circ h$ *entón* $g = h$,
- $f : a \to b$ *é* isomorfismo *se existe un morfismo* $f^{-1} : b \to a$, *tal que* $f \circ f^{-1} = 1_b, f^{-1} \circ f = 1_a$. f^{-1} *é o morfismo inverso de* f, *e está determinado de forma única por* f. *Se un morfismo ten inverso, este tamén é isomorfismo e o seu inverso é o isomorfismo dado.*

A categoría de conxuntos $\mathcal{S}et$ é unha categoría cuios obxectos son os conxuntos, os morfismos entre os obxectos a, b son as aplicacións entre eses conxuntos e o morfismo identidade dun obxecto, é a aplicación identidade dese conxunto.

Na categoría de conxuntos os morfismos épicos son as aplicacións sobrexectivas, os morfismos mónicos son as aplicacións inxectivas e os isomorfismos as aplicacións bixectivas.

Se Σ é unha signatura, as Σ-álxebras como obxectos, e os Σ-homomorfismos como morfismos forman unha categoría denominada *categoría de* Σ*-álxebras*, Alg_Σ.

Se $(\mathcal{A}, \circ)$ é unha categoría e a é un obxecto de $\mathcal{A}$, obtense unha nova categoría $a/_{\mathcal{A}}$ (*categoría co-slice*) cuios obxectos son os morfismos de dominio a e se $f : a \to b$, $g : a \to c$ son dous obxectos, un morfismo h en $a/_{\mathcal{A}}$ de f a g vén dado por $h : b \to c, h \circ f = g$.



Definición 2.34 *Dada unha categoría* $(\mathcal{A}, \circ, 1)$, $\mathcal{B}$ *é unha* subcategoría *de* $\mathcal{A}$ *se* $(\mathcal{B}, \circ, 1)$ *forma unha categoría, onde os obxectos e morfismos en* $\mathcal{B}$ *son a súa vez, obxectos e morfismos en* $\mathcal{A}$.

A categoría de conxuntos finitos $\mathcal{F}in\mathcal{S}et$ é unha subcategoría de $\mathcal{S}et$.

Definición 2.35 *Un* obxecto inicial *nunha categoría é un obxecto* s *tal que para todo outro obxecto* a *existe un único morfismo* $f : s \to a$.

O coproducto *dos obxectos* a *e* b *da categoría* $\mathcal{A}$, *é un obxecto* $a + b$ *e un par de morfismos* $j_a : a \to a + b$, $j_b : b \to a + b$ *tal que para calquera outro obxecto* c *e morfismos* $f : a \to c$, $g : b \to c$ *existe un único morfismo* $< f, g >: a + b \to c$ [8] *que verifica* $f =< f, g > \circ j_a$ *e* $g =< f, g > \circ j_b$. *O coproducto dun conxunto* S *de obxectos representarase por* $\coprod_{a \in S}\{a\}$

Sexan $f, g : a \to b$. *Un* coigualador *de* f, g *é un morfismo* $h : b \to c$ *tal que :*

$h \circ f = h \circ g$,

dado calquera outro morfismo $k : b \to d$ *tal que* $k \circ f = k \circ g$ *existe un único morfismo* $l : c \to d$ *e que* $l \circ h = k$.

O push-out *dos morfismos* $f : a \to b$, $g : a \to c$ *é un obxecto* d, *e dous morfismos* $h : b \to d, k : c \to d$ *tal que para calquera outro obxecto* d' *e morfismos* $h' : b \to d', k' : c \to d'$ *existe un único morfismo* $l : d \to d'$ *que verifica* $l \circ h = h', l \circ k = k'$. *o* par co-núcleo *de* f *é o push-out de* (f, f).

As definicións duais[9] *de obxecto inicial, coproducto, coigualador, push-out, par co-núcleo, denomínanse respectivamente* obxecto final, producto, igualador, pull-back, par-núcleo.

Un obxecto cero *é un obxecto que é inicial e final.*

2.41 *En calquera categoría* $\mathcal{A}$ *verifícase que :*

o obxecto inicial se existe, é único agás isomorfismos,

o coproducto de dos obxectos, se existe, é único salvo isomorfismos,

o push−out de dos morfismos, se existe, é único agás isomorfismos.

Proba. Probarase a unicidade do coproducto. Sexan a, b dous obxectos e dous coproductos $(a + b, j_a, j_b)$, $((a + b)', j'_a, j'_b)$. Pola definición, existe un morfismo $h : a + b \to (a + b)'$, $h \circ j_a = j'_a$, $h \circ j_b = j'_b$. Pero, tamén por definición, existe $h' : (a+b)' \to a+b$, $h' \circ j'_a = j_a$, $h' \circ j'_b = j_b$. Por elo, e aplicando de novo a definición

[8] a notación usual $[f, g]$ non é utilizada, para reservar esta notación para o reemplazamento.

[9] dado un morfismo $f : a \to b$ o seu dual é $f : b \to a$. Dada calquera categoría, constrúese a categoría dual cos seus mesmos obxectos e cos morfismos duais dos de aquela. A nocion dual dunha dada obtense cambiando os morfismos enpregados por seus duais.

de coproducto, a composición $h' \circ h$ e a identidade 1_{a+b}, son iguais por verificar que $j_a = h' \circ h \circ j_a$, $j_b = h' \circ h \circ j_b$ e $j_a = 1_{a+b} \circ j_a =$, $j_b = 1_{a+b} \circ j_b$. Por elo, $h' \circ h = 1_{a+b}$. De forma análoga probase que $h \circ h' = 1_{(a+b)'}$ □.

2.42 *Os coigualadores son morfismos épicos e os igualadores son mónicos.*

Definición 2.36 *Sexa C unha categoría. Diráse que $f : a \to b$ ten unha* factorización coigualador-mono *se existen dous morfismos f' e e tales que : f' é un coigualador, e é mónica e $f = e \circ f$.*

A categoría ten factorización coigualador-mono, se a ten todo morfismo da categoría.

A descomposición coigualador-mono é evidentemente unha condición máis forte que a simple esixencia de que o morfismo f' sexa épico e e mónico. A descomposición coigualador-mono é única agás isomorfismos, o que non sucede, en xeral, coa expresión dun morfismo como a composición dunha épica e outra mónica. Nas categorías nas que existe unha factorización coigualador-mono os morfismos conxuntamente épicos e mónicos son isomorfismos.

A categoría de conxuntos ten obxecto inicial e obxecto final (aínda que non obxecto cero), productos e coproductos, igualadores e coigualadores, pull-backs e push-outs e factorización coigualador-mono.

Definición 2.37 *Sexan A, B dúas categorías. Un* funtor *de A a B, é un par (F_{ob}, F_{mor}) de funcións que verifican :*

para todo obxecto a de A, $F_{ob}(a)$ é un obxecto de B,

para todo morfismo $f : a \to b$ de A, $F_{mor} : F_{ob}(a) \to F_{ob}(b)$ é un morfismo de B,

$F_{mor}(f * g) = (F_{mor}(f)) \circ (F_{mor}(g))$[10],

$F(1_a) = 1_{F(a)}$

Se F é un funtor, representarase por F tanto a súa actuación nos obxectos como nos morfismos.

As categorías son, tamén os obxectos doutra categoría Cat cuios morfismos son os funtores entre categorías. A identidade en Cat para a categoría C é o funtor identidade $1_C : C \to C$ que leva todo obxecto e todo morfismo en sí mesmo.

Definición 2.38 *Sexan A, B categorías e $F : A \to B$, $G : B \to A$ funtores. F é* adxunto *pola esquerda de G ($F \dashv G$) se :*

> *para cada par (a, b) de obxectos, a de A e b de B existe unha bixección ϕ entre os conxuntos $A[a, Gb]$ e $B[Fa, b]$ tal que :*

[10]onde $*$ e $\circ$ son as composicións en A e B respectivamente.

- *para todo* $f, g, \alpha\ f : a' \to a, g : b \to b', \alpha \in A[a, Gb]$, $\phi(\alpha * f) = \phi(\alpha) \circ F(f)$ *e* $g \circ \phi(\alpha) = \phi(G(g) * \alpha)$.

Se $f : a \to G(b)$ é un morfismo de A, o adxunto de f, ó morfismo $\phi(f)$ de B. Igualmente se $g : F(a) \to b$ é un morfismo de B, o adxunto de g, é $\phi^{-1}(g)$.

O morfismo $\phi^{-1}(1_{F(a)}) : a \to G(F(a))$ denomínaselle *unidade da adxunción* en a e represéntase por η_a. A *counidade da adxunción* en b, ε_b é o morfismo $\phi(1_{G(b)}) : F(G(b)) \to b$.

Definición 2.39 *Sexan* $F, G : A \to B$ *funtores. Unha* transformación natural τ *entre* F *e* G*, é unha regra que asigna a todo obxecto* a *de* A*, un morfismo* $\tau_a : F(a) \to G(A)$ *tal que para todo* $f : a \to b$ *morfismo en* A*, verifícase o diagrama conmutativo :*

$\tau_b \circ F(f) = G(f) \circ \tau_a$

$$\begin{array}{ccc} F(a) & \xrightarrow{F(f)} & F(b) \\ {\scriptstyle \tau_a}\downarrow & & \downarrow{\scriptstyle \tau_b} \\ G(a) & \xrightarrow[G(f)]{} & G(b) \end{array}$$

Se $F \dashv G$, a unidade $\eta : 1_A \to F$ e a counidade $\varepsilon : G \to 1_B$ son transformacións naturais.

2.3.2 Teorías alxebraicas

Definición 2.40 *Unha* teoría alxebraica *nunha categoría* K*, é un triple* (T, η, μ) *onde* T *é un funtor de* K *en* K*,* η *é unha transformación natural do funtor identidade de* K *en* T *e* μ *unha transformación natural de* T^2 *en* T*, tal que para todo obxecto* a *de* K *os diagramas conmutan :*

$$\begin{array}{ccc} T^3(a) & \xrightarrow{T(\mu_a)} & T^2(a) \\ {\scriptstyle \mu_{T(a)}}\downarrow & & \downarrow{\scriptstyle \mu_a} \\ T^2(a) & \xrightarrow[\mu_a]{} & T(a) \end{array} \qquad \begin{array}{ccc} T(a) & \xrightarrow{T(\eta_a)} & T^2(a) \\ & {\scriptstyle id_{T(a)}}\searrow & \downarrow{\scriptstyle \mu_a} \\ & & T(a) \end{array} \qquad \begin{array}{ccc} T(a) & \xrightarrow{\eta_{T(a)}} & T^2(a) \\ & {\scriptstyle id_{T(a)}}\searrow & \downarrow{\scriptstyle \mu_a} \\ & & T(a) \end{array}$$

Outras caracterizacións das teorías alxebraicas pódense topar no traballo de Fernandez e Freire [65].

Definición 2.41 *[117] Sexa* (T, η, μ) *unha teoría alxebraica en* K*. A categoría* K_T *cuios obxectos son os de* K*, e dados dous obxectos* A *e* B*, os seus morfismos* $f : A \to B$ *veñen dados por morfismos en* K *da forma* $A \to T(B)$*. A composición en* K_T *de* $A \to B$ *con* $B \to C$ *obtense por :* $A \xrightarrow{\mu_C \circ T(g) \circ f} T(C)$*. Esta categoría toma o nome de* categoría de Kleisli.

2.43 *Sexa* Σ *unha signatura. O triple* (T_Σ, η, μ) *dado polas seguintes condicións é unha teoría alxebraica :*

- $T_\Sigma : Set \longrightarrow Set$, *é o funtor que leva :*

  se X *é un conxunto* $T_\Sigma(X)$ *é o conxunto de termos de variables en* X,

  se $f : X \rightarrow Y$ *é unha aplicación ,* $T_\Sigma(X) \overset{T_\Sigma(f)}{\rightarrow} T_\Sigma(Y)$ *é a aplicación que sustitúe cada elemento* x *de* X *en cada termo* $t[x]$ *por* $f(x)$.

- $\eta_X : X \rightarrow T_\Sigma(X)$ *dada por* $\eta_X(x) = x$, *ou máis concretamente* $\eta_X(x) = \{(<>, x)\}$,

- $\mu_X : T_\Sigma(T_\Sigma(X)) \rightarrow T_\Sigma(X)$ *dada pola eliminación dos parénteses, esto é,* $\mu_X(\{(<>, t)\}) = t$, *e* $\mu_X(f(t_1, \ldots, t_n)) = f(\mu_X t_1, \ldots, \mu_X t_n)$. *Esta operación transforma as follas de* $T_\Sigma(T_\Sigma(X))$ *etiquetadas polos elementos de* $T_\Sigma(X)$ *en nós, xeralmente non terminais.*

Substituíndo na propiedade anterior, T_Σ por $T_{\Sigma,E}$ onde este é o funtor que leva todo conxunto X na álxebra de termos en X da teoría dada por E, obtense igualmente unha categoría de Kleisli.

2.3.3 Categoría de substitucións

A categoría de Kleisli correspondente ó triple (T_Σ, η, μ) ten unha subcategoría de interese especial, aquela na que os obxectos son conxuntos finitos.

Os morfismos $\sigma : X \rightarrow Y$ correspóndense de xeito unívoco con aplicacións $\sigma : X \rightarrow T_\Sigma(Y)$, e polo tanto, un morfismo nesa categoría da lugar a unha substitución $\sigma^\# : T_\Sigma(X) \rightarrow T_\Sigma(Y)$ resultado de reemplazar as variables de X polas imaxes mediante σ. Por elo, esta subcategoría chamarase tamén *Categoría de substitucións* ou $Catsubst$.

No caso dunha teoría ecuacional, dada por E, a subcategoría formada ten como morfismos as clases de substitucións que se corresponden cos homomorfismos $T_E(X) \rightarrow T_E(Y)$. Esta categoría será expresada por $FinKleisli_E$, e $Catsubst$ é xustamente $FinKleisli_\emptyset$. No que sigue, a notación $FinKleisli$ será empregada para unha categoría arbitraria $FinKleisli_E$.

A substitución $\sigma^\# = \mu_Y \circ T_\Sigma(\sigma)$ expresa o homomorfismo universal dado polo carácter libre de $T_\Sigma(X)$ (análogamente para o caso ecuacional).

Se X non é finito, os morfismos $\sigma : X \rightarrow Y$ equivalentes ás aplicacións $\sigma : X \rightarrow T_\Sigma(Y)$ non se corresponden necesariamente con substitucións, xa que un morfismo pode mover un número de elementos de X non finito.

Lémbrase que ó utilizar substitucións en vez de sustitucións esixe restrinxir o dominio de actuación de éstas. Así, non se pode mover nengunha variable que non se encontre no conxunto finito de partida e as variables da imaxe están determinadas explícitamente, xa que están incluídas no conxunto finito de chegada. Por elo, dada unha substitución σ non comprirá empregar a "protección de variables" que consiste en seleccionar un conxunto W de variables, tal que $Dom(\sigma) \cup Im(\sigma) \subseteq W$. Cando se utiliza unha regra ecuacional, as súas variables deben destar fora de dito conxunto protector.

No caso de teorías de *Xénero Ordenado (Order Sorted)* o control das variables aínda se fai máis preciso, como se mostra no traballo de Meseguer *et al.* [124].

Dada unha *substitución*, existe unha única *sustitución* que se corresponde con ela, pero o recíproco non é certo, xa que unha *substitución* $\sigma' : X \rightharpoonup Y$ obtense da *sustitución* σ seleccionando calesquera X, Y finitos tales que $Dom(\sigma) \subseteq X, I(\sigma) \subseteq Y$, $x\sigma' = x$ se $x \notin Dom(\sigma)$.

Na maioría dos casos, as sustitucións e as substitucións compórtanse en forma idéntica, se ben existen propiedades dunha sustitución que non implican a mesma propiedade en tódalas substitucións correspondentes.

Sexa σ sustitución idempotente, de $Dom(\sigma) = X$; $I(\sigma) = Y$. A correspondente substitución $\sigma' : X \rightharpoonup Y$ non é idempotente, xa que para que estea definida a sua composición, debe verificarse que $X = Y$ pero esto só se cumple na identidade $X = Y = \emptyset$. Pode seleccionarse unha idempotente correspondente a σ, escollendo $X = Y = Var(\sigma) = Dom(\sigma) \cup I(\sigma)$.

A inclusión expresarase na forma $j_X : X \rightharpoonup Y$ (ou ben $j_{X,Y}$ se se quere expresar o novo conxunto), o coproducto $X + Y$ e a universalidade do coproducto para $X \xrightarrow{\sigma} Z, Y \xrightarrow{\tau} Z$ por $< \sigma, \tau >$. Por outra parte, para diferenciar a aplicación dunha substitución σ nun termo j, da composición de j_Y con σ, esta última represéntase explícitamente por $j_Y \circ \sigma$.

A seguinte propiedade, enunciada para *Catsubst* verifícase igualmente para toda categoría de Kleisli como consecuencia da unicidade de $\sigma^{\#}$ tal que $\sigma^{\#}\eta_X = \sigma$.

2.44 *Catsubst ten obxecto inicial, e coproductos finitos.*

Xa se probou o carácter inicial de $\mathcal{T}(\emptyset)$ respecto de $\mathcal{T}(X)$. Para a existencia do coproducto : sexan X e Y dos conxuntos de variables e Y' outro conxunto, tal que $|Y| = |Y'|$ e $X \cap Y' = \emptyset$. De aí que $\exists \rho : Y \to Y'$ unha permutación. Defínese $X + Y = X \cup Y'$, $j_X : X \rightharpoonup X + Y$, $j_Y : Y \rightharpoonup X + Y$ dadas por :

- j_X vén dada pola composición (en *Set*) de $X \to X \cup Y'$ con $\eta_{X \cup Y'}$,
- j_Y é a composición de ρ coa inclusión $Y' \to X \cup Y'$ composta de novo con $\eta_{X \cup Y'}$.

Para probar a universalidade do coproducto : Sexan $\sigma : X \rightharpoonup Z$, $\tau : Y \rightharpoonup Z$. Defínese $< \sigma, \tau > : X + Y \rightharpoonup Z$ por $x < \sigma, \tau > = x\sigma$ se $x \in X$, $y < \sigma, \tau > = (y\rho^{-1})\tau$. Está ben definida e : $j_X \circ < \sigma, \tau > = \sigma$, $j_Y \circ < \sigma, \tau > = \tau$. A unicidade obtense do feito de que toda variable de $X + Y$ é de X o de Y', e por elo debe ser levada necesarimente por σ se é de X ou por $\rho^{-1}\tau$ se é de Y' para garantir a conmutatividade do diagrama $\square$.

Este resultado é igualmente certo para toda *FinKleisli*, como consecuencia do resultado xa mencionado de Birkoff acerca da existencia da álxebra libre xerada por X en toda variedade ecuacional.

Mónicas, Epicas e Isomorfismos en Catsubst

Neste apartado, mostramos caracterizacións tanto dos morfismos nunha categoría $FinKleisli$ como as particularizacións en $Catsubst$.

A seguiente proposición caracteriza os isomorfismos en $FinKleisli$.

2.45 *Sexa $\sigma : X \rightharpoonup Y$. E isomorfismo sse $\sigma^{\#} : T(X) \to T(Y)$ é bixectiva.*

Proba $\Rightarrow$. A substitución $X \xrightarrow{\sigma} Y$ ten como inversa a $Y \xrightarrow{\sigma^{-1}} X$, polo que a partir das aplicacións $X \xrightarrow{\sigma} T(Y)$, $Y \xrightarrow{\sigma^{-1}} T(X)$ entón $\mu_X T(\sigma^{-1})\sigma = \eta_X$ e $\mu_Y T(\sigma)\sigma^{-1} = \eta_Y$. Xa que $\sigma = \sigma^{\#}\eta_X$, dedúcese que $(\sigma^{-1})^{\#}\sigma^{\#} = 1_{T(X)}$ pola unicidade e análogamente $\sigma^{\#}(\sigma^{-1})^{\#} = 1_{T(Y)}$.

$\Leftarrow$. Sexa $\sigma^{\#}$ bixectiva. Existe polo tanto $(\sigma^{\#})^{-1} : T(Y) \to T(X)$. Defínese $\sigma' = (\sigma^{\#})^{-1}\eta_Y$. Hai que probar que $Y \xrightarrow{\sigma'} X$ é a inversa de $X \rightharpoonup Y$.

Primeiro probarase que, dadas $\alpha : X \to T(Y), \beta : Y \to T(Z)$ calesquera, cúmplese que $(\beta\alpha)^{\#} = \beta^{\#}\alpha^{\#}$. Esto é certo posto que $(\beta\alpha)^{\#} = \mu_Z T(\beta^{\#}\alpha) = \mu_Z(T(\beta))^{\#}T(\alpha) = \beta^{\#}\mu_Y T(\alpha) = \beta^{\#}\alpha^{\#}$ (usando a naturalidade de μ e a funtorialidade de T).

En segundo lugar probarase que $(\sigma')^{\#} = (\sigma^{\#})^{-1}$. Aplicando a propiedade anterior, tense que $(\sigma^{\#}\sigma')^{\#} = \sigma^{\#}(\sigma')^{\#} = \eta^{\#} = 1_{T(Y)}$. Polo carácter bixectivo de $\sigma^{\#}$, verifícase $(\sigma')^{\#} = (\sigma^{\#})^{-1}$.

Por último probarase que $\sigma'\sigma = 1_X$, $\sigma\sigma' = 1_Y$. Esta composición en Set corresponde a $\sigma^{\#}\sigma'\eta_Y = \sigma^{\#}(\sigma^{\#})^{-1}\eta_Y$. Para a primeira, $(\sigma')^{\#}\sigma = (\sigma^{\#})^{-1}\sigma = (\sigma^{\#})^{-1}\sigma^{\#}\eta_X = \eta_X$, e polo tanto tamén se ten que $\sigma'\sigma = 1_X$ en $FinKleisli$ □.

De onde se deduce que, no caso de $Catsubst$, trátase dunha permutación, dada a finitariedade de X, Y e que las variables deben ser alcanzadas.

A seguinte propiedade caracteriza os morfismos mónicos.

2.46 *$\sigma : X \rightharpoonup Y$ é mónica sse a correspondente $\sigma^{\#} : T(X) \to T(Y)$ é inxectiva.*

Proba $\Rightarrow$. Sexan $Z \underset{\rho}{\overset{\theta}{\rightrightarrows}} T(X) \xrightarrow{\sigma^{\#}} T(Y)$ tal que $\sigma^{\#}\theta = \sigma^{\#}\rho : Z \to T(Y)$. Pasando a $FinKleisli$ tense $\theta \circ \sigma = \rho \circ \sigma$ de onde $\theta = \rho$. Para a outra implicación o razonamento é semellante □.

O anterior resultado, en $Catsubst$ e a nivel de variables e termos significa que unha substitución σ é mónica sse : a propiedade $NM(x\sigma, x)$, dada pola definición recursiva que segue, non se verifica para nengunha variable x, tal que $x\sigma \neq x$:

$NM(s,t)$ se se cumple algunha das condicións :

- $t \in X$, $s = t\sigma$ e unha das seguintes condicións é verificada
  - existe $y \in X, s = y\sigma$, $y \neq t$,
  - $s = t$,
  - $s = f(s_1, \ldots, s_n)$, e verifícase que existe $\{t_i | 1 \leq i \leq n\}$, tal que $NM(s_i, t_i)$ se cumple para todo i, $1 \leq i \leq n$,

- $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, e para todo i, $1 \leq n$, $NM(s_i, t_i)$. Para $n = 0, NM(a, a)$.

Sexan dous termos $u, v, u \neq v$ que son levados por $\sigma^{\#}$ nun mesmo elemento, probarase que existe algún $x \in X$, que verifica $NM(x\sigma, x), x \neq x\sigma$. Esto obtense aplicando inducción no número de operadores de u e v.

- se $u = x$, e $v = y$ entón $NM(x\sigma, x)$,
- se $u = x, v = f(v_1, \ldots, v_n)$ entón necesariamente $x\sigma = f(u_1, \ldots, u_n)$. Xa que $v\sigma = f(v_1\sigma, \ldots, v_n\sigma)$, obtense $NM(x\sigma, x)$ xa que para todo $1 \leq i \leq n$, $NM(v_i\sigma, v_i)$,
- noutro caso, $u = f(u_1, \ldots, u_n)$, $v = f(v_1, \ldots, v_n)$ onde algún $u_i \neq v_i$ polo que $MN(x\sigma, x), x\sigma \neq x$ por inducción.

Por outra parte, de verificarse $NM(x\sigma, x)$, $x\sigma \neq x$ dedúcese a existencia dun termo $v \neq x$ que ten a mesma imaxe por σ que x, o que se obtén por inducción aplicando a definición de NM.

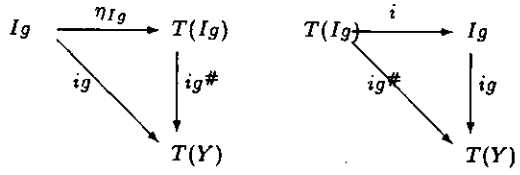
Pola relación $\cong$, expresarase a igualdade agás isomorfismos en $FinKleisli$. Por elo, no caso da teoría libre, $X \cong Y$ sse existe unha permutación entre ambos, e polo tanto, deben de ter o mesmo número de elementos. No caso xeral, esto non se verifica. E.g. Na teoría dada pola ecuación $x =_E y$ só existe un único termo e dados calesquera conxuntos finitos X, Y existen isomorfismos entre X e Y.

Definición 2.42 *Sexa $\sigma : X \rightharpoonup Y$. Dise que $\sigma' : X \rightharpoonup Y'$ é unha factorización non trivial de σ se $Y \cong Y' + Y''$, $\sigma' \circ j_{Y'} = \sigma$, $Y \ncong Y'$. A menor factorización $\sigma' \circ j_{Y'}$ con σ' minimal no orden $\leq$ chamarase* factorización minimal *de σ.*

Para caracterizar as épicas desenrólase un resultado previo.

2.47 *Sexan $Y \overset{\theta}{\underset{\gamma}{\rightrightarrows}} Z$, e sexa $(Ig, Ig \xrightarrow{ig} T(Y))$ o igualador en Set de $\theta^{\#}, \gamma^{\#} : T(Y) \rightrightarrows T(Z)$. Entón $T(Ig) \cong Ig$. Amais, se $\theta \neq \gamma$ entón Ig está estrictamente contido en $T(Y)$.*

Proba. Primeiro demóstrase que $ig^{\#} : T(Ig) \to T(Y)$ iguala a $\theta^{\#}, \gamma^{\#}$. Esto é certo pola unicidade de $\theta^{\#} ig^{\#} : T(Ig) \to T(Z)$ dado o carácter libre de $T(Ig)$. De forma que existe $i : T(Ig) \to Ig$ que fai conmutativo os diagramas



$ig = ig^{\#}\,\eta_{Ig}$, $ig\,i = ig^{\#}$.

Xa que $ig = ig\,i\,\eta_{Ig}$ e, por ser igualador, ig é mónica, tense que $i\,\eta_{Ig} = 1_{Ig}$.

Para probar que $\eta_{Ig}\,i = 1_{T(Ig)}$ emprégase que η_{Ig} leva Ig en $T(Ig)$ mediante : $\eta_{Ig}\,i\,\eta_{Ig} : Ig \to T(Ig) \to Ig \to T(Ig)$ cun único $(\eta_{Ig}\,i\,\eta_{Ig})^{\#} : T(Ig) \to T(Ig)$. Pero, xa que $i\,\eta_{Ig} = 1_{Ig}$, tense polo tanto que $(\eta_{Ig}\,i\,\eta_{Ig})^{\#} = \eta_{Ig}^{\#} = 1_{T(Ig)}$, e por elo $Ig \cong T(Ig)$.

Para a segunda parte sexan $Y \overset{\theta}{\underset{\gamma}{\rightrightarrows}} T(Z)$. No caso de que $Ig \simeq T(Y)$ entón $Y \overset{\eta_Y}{\to} T(Y) \overset{ig}{\to} T(Y) \overset{T(\theta)}{\underset{T(\gamma)}{\rightrightarrows}} T^2(Z) \overset{\mu_Z}{\to} T(Z)$ pero de aquí resulta unha única substitución $\theta = \gamma : Y \to T(Y) \to T(Z)$ pola construcción do igualador en $\mathcal{S}et$[11], o que contradí a hipótese. □.

2.48 *$\sigma : X \rightharpoonup Y$ épica sse non existe unha factorización non trivial de σ.*

Proba $\Rightarrow$. Supoñendo que existe unha factorización non trivial de $\sigma = \sigma' \circ j_{Y'}$ onde $Y = Y' + Y''$. Sexan $Y'' \overset{\theta}{\underset{\gamma}{\rightrightarrows}} Z$, $\lambda : Y' \rightharpoonup Z$, con $\theta \neq \gamma$. Polo tanto $Y \overset{<\lambda,\theta>}{\underset{<\lambda,\gamma>}{\rightrightarrows}} Z$, verifican : $\sigma\ <\lambda,\theta> = \sigma' \circ j_{Y'} \circ <\lambda,\theta> = \sigma'\lambda = \sigma' \circ j_{Y'} \circ <\lambda,\gamma> = \sigma\ <\lambda,\gamma>$, de onde se deduce que σ non é épica, xa que $<\lambda,\gamma> \neq <\lambda,\theta>$.

$\Leftarrow$. Sexa σ non épica. Entón existen : $Y \overset{\theta}{\underset{\gamma}{\rightrightarrows}} Z, \sigma\theta = \sigma\gamma$, con $\theta \neq \gamma$. Pasando a Set, tense $X \to Im(\sigma) \to T(Y)$ factorización coigualador-mono de σ. O igualador de $\theta^{\#}, \gamma^{\#}$ é da forma $T(Y') \to T(Y)$, e amais $Im(\sigma) \to T(Y')$ (único) xa que σ iguala a ese par, $T(Y') \neq T(Y)$ posto que $\theta \neq \gamma$, entón $X \to Im(\sigma) \to T(Y')$ e ésta corresponde a unha factorización non trivial de σ □.

Unha substitución $\sigma : X \rightharpoonup Y$ é épica se tódalas variables de Y son alcanzadas por σ, i.e. $I(\sigma) = Y$. E claro que se unha variable $y \in Y, y \notin I(\sigma)$, existen $\theta \neq \gamma$, $\sigma\theta = \sigma\gamma$. Se non é épica, existe $t \in T_Y$, $t\theta \neq t\gamma$ anque para todo $x \in X, x\sigma\theta = x\sigma\gamma$. O resultado obtense directamente de aplicar inducción no tamaño de t.

En *Catsubst* mónica e épica non é necesariamente un isomorfismo.

Exemplo 2.9 *E.g.* $X = \{x\}, Y = \{y\}$, $x\sigma = f(y)$.

2.49 *En FinKleisli todo morfismo pódese expresar como $\sigma'\sigma''$, onde σ' é epica e σ'' mónica.*

Proba. Sexa $\sigma : X \rightharpoonup Y$ unha substitución. Se σ é épica, entón $(\sigma, 1_Y)$ é a composición epi-mono buscada. Se σ non é épica, existe unha factorización non trivial $(\sigma', j_{Y'})$, onde $|Y'| < |Y|$. Se σ' é épica, xa está obtida a descomposición epi-mono, dado que $j_{Y'}$ é mónica. Se nono é, e posto que Y é finito, obtense unha sucesión $(\sigma_0, 1_Y), (\sigma_1, j_{Y_1}), \ldots, (\sigma_n, (j_{Y_n} \circ \ldots j_{Y_1}))$. O último elemento desa sucesión está formado por unha parte épica $\sigma' = \sigma_n$ e unha mónica $\sigma'' = j_{Y_n} \circ \ldots j_{Y_1}$ □.

[11] na categoría de conxuntos, a construcción clásica do conxunto igualador é $Ig = \{x | f(x) = g(x)\}$, $ig(x) = x$.

A factorización epi-mono permite eliminar as variables sobrantes, esto é, aquelas que non teñen sido alcanzadas por nengunha variable do conxunto de partida. No que segue, se dirá parte épica de unha substitución σ á substitución σ' tal que $\sigma' \circ j_X = \sigma$.

En *Catsubst*, verifícase que se $\sigma\lambda = \tau$, e denotando por σ_{ep}, τ_{ep} as partes épicas de σ e τ respectivamente, existe λ', $\sigma_{ep}\lambda' = \tau_{ep}$. Esto é consecuencia de que para toda variable $z \in cod(\tau_{ep})$ existe $x \in dom(\sigma)$, $x\sigma\lambda = x\tau$, $z \in Var(x\tau)$. Sexa $\pi \in \Pi_z(x\tau)$. Se $\pi = <>$, entón $x\sigma = y \in cod(\sigma_{ep})$, $y\lambda = z$. Noutro caso, se $\pi \in \Pi_{x\sigma}$ dase unha situación semellante ó caso anterior. Se $\pi \notin \Pi_{x\sigma}$, existe $y \in cod(\sigma_{ep})$, $z \in Var(y\lambda)$. De feito, se $Y = cod(\sigma_{ep})$, $\lambda' = ((j_Y \circ \lambda)_{ep})_{ep}$.

Por outra parte, non sempre $(j_Y \circ \lambda)_{ep} = j_Y \circ \lambda_{ep}$ como se observa en :

Exemplo 2.10 *Sexan* $\sigma : \{i\} \rightharpoonup \{x,y\}, \tau : \{i\} \rightharpoonup \{z,u\}, \lambda : \{x,y\} \rightharpoonup \{z,u\}$, *dadas por :* $i\sigma = f(x)$, $x\lambda = f(z)$, $y\lambda = f(u), i\tau = f(f(z))$ *tense que* $\lambda' : \{x\} \rightharpoonup \{z\}$, $x\lambda' = f(f(z))$, $j_{\{x\}} \circ \lambda_{ep} : \{x\} \rightharpoonup \{z,u\}$

2.50 *Sexa* $\sigma : X \rightharpoonup X$, σ *é idempotente sse* $\sigma = <1_{X'}, \sigma'> j_{X'} : X' + X'' \to X$.

Proba $\Rightarrow$. Se σ é idempotente debe tomar a forma $X \rightharpoonup X$ para que estea definida a composición. Sexa $X' + X'' = X$, $X'' = Dom(\sigma)$. Por construcción $\sigma = <1_{X'}, \sigma'>$, faise necesario que $I(\sigma) \subseteq X'$ para que sexa idempotente. Polo tanto a factorización minimal de σ ten a forma $\sigma'' \circ j_{X'}$ xa que tódalas variables de X' son trivialmente alcanzadas por sí mesmas.

$\Leftarrow$. Por definición de coproducto, σ vén caracterizada de forma única pola súa composición con $j_{X'}$ e con $j_{X''}$ dadas por $j_{X'} \circ \sigma$ e por $j_{X''} \circ \sigma = \sigma' \circ j_{X'}$.

Pero de ésto se obtén que :

$$j_{X'} \circ \sigma\sigma = j_{X'} \circ <1_{X'}, \sigma'> \circ j_{X'} \circ \sigma = j_{X'} \circ \sigma,$$

$$j_{X''} \circ \sigma\sigma = j_{X''} \circ <1_{X'}, \sigma'> \circ j_{X'} \circ \sigma = \sigma' \circ j_{X'} \circ <1_{X'}, \sigma'> \circ j_{X'} = \sigma' \circ j_{X'} = j_{X''} \circ \sigma$$

polo tanto σ é idempotente □.

Amais da descomposición epi-mono de toda substitución, existe outra forma máis forte de romper unha substitución mediante a extracción dos primeiros operadores.

2.51 *Sexa* $\sigma : \{i\} \rightharpoonup Y$ *unha substitución. Se* σ_{ep} *non é isomorfismo, entón existe unha factorización* $(\bar{\sigma}, \underline{\sigma})$ *de* σ *verificando a seguintes condicións :*

- $\bar{\sigma}\underline{\sigma} = \sigma$,
- $\bar{\sigma}$ *é épica pero non é isomorfismo,*
- *dado calquera outro par* (θ, θ') *tal que* $\theta\theta' = \sigma$ *ou ben a parte épica de* θ *é isomorfismo, ou ben* $\bar{\sigma} \leq \theta$.

Proba : Se $i\sigma$ é variable, claramente σ é isomorfismo. Noutro caso, sexa $i\sigma = f(t_1, \ldots, t_n)$. Pode definirse $\bar{\sigma} : \{i\} \rightharpoonup Z = \{z_1, z_2, \ldots, z_n\}$, $\underline{\sigma} : X \rightharpoonup Y$ dadas por $i\bar{\sigma} = f(z_1, \ldots, z_n)$ y $z_i\underline{\sigma} = t_i$. E claro que se verifican as tres condicións $\square$. Esta factorización chamarase *factorización do primeiro operador.*

A anterior propiedade de *Catsubst*, non se verifica en xeral en *FinKleisli.*

Exemplo 2.11 $f(g(x)) =_E h(g(x))$, *a substitución* $\{i\} \xrightarrow{\sigma} X$, *ten dúas factorizacións* $\{i\} \xrightarrow{\sigma_1} \{y\} \xrightarrow{\sigma_2} \{x\}$, $\{i\} \xrightarrow{\sigma_1'} \{y\} \xrightarrow{\sigma_2'} \{x\}$, *dadas por* $i\sigma_1 = f(y)$, $y\sigma_2 = g(x)$, $i\sigma_1' = h(y), \sigma_2' = \sigma_2$.

Como consecuencia da propiedade anterior :

2.52 *Sexan* $\sigma_i : \{x\} \rightarrow Y$, $i = 1, 2$ *substitucións épicas, non isomorfismos e unificables. Necesariamente* $\bar{\sigma}_1 \equiv \bar{\sigma}_2$.

Dado $\sigma : \{x_1, \ldots, x_m\} \rightharpoonup Y$, cabe agardar que se obteña un resultado semellante con $\bar{\sigma} = <\bar{\sigma}_1, \bar{\sigma}_2, \ldots, \bar{\sigma}_m>$, $\sigma_i = j_{\{x_i\}} \circ \sigma$. Esto, non se verifica como se observa para

Exemplo 2.12 $X = \{x, y\}$, $x\sigma = f(z)$, $y\sigma = g(u)$. *Téñense substitucións épicas non isomorfismos* $\theta = <f, 1_{\{y\}}>$, $\delta = <1_{\{x\}}, g>$, *tales que* $\theta \not\leq \delta \not\leq \theta$, $\theta = j_{\{x\}}\bar{\sigma}_{\{x\}}$, $\delta = j_{\{y\}}\bar{\sigma}_{\{x\}}$.

Para aproveitar a descomposición das substitucións e poder empregar unha notación semellante á dos termos, introdúcense as posicións dunha substitución dada, así como a substitución correspondente a esa posición. Para elo, supónse dado un orden total no conxunto das variables.

No que segue, se $X = \{x_1, \ldots, x_m\}$ é un conxunto finito de variables, e x_i expresará a i-ésima variable do conxunto co orden dado previamente, identificando xeralmente x_i con i. Obsérvese que a substitución $\{1, \ldots, n\} \xrightarrow{\sigma} Y$ pódese expresar de forma única (seguindo o orden dado) como $<\sigma_1, \ldots, \sigma_n>$, $\{i\} \xrightarrow{\sigma_i} Y$.

Definición 2.43 *Sexa* $I \xrightarrow{\sigma} X$ *unha substitución. Dise que* π *é unha posición de* σ, *en cuio caso* σ/π *é a substitución correspondente á posición* π *de* σ *se :*

- *se* $\pi = <>$, $\sigma/\pi = \sigma$,
- *se* $\pi = i \cdot \omega$ *onde :*
  - *se* $|I| = 1$ *e* $\{i\} \xrightarrow{f} Y \xrightarrow{\sigma'} X$ *é a factorización do primeiro operador para* σ, *entón* ω *é unha posición de* σ' *e* $\sigma/\pi = \sigma'/\omega$
  - *se* $|I| = n > 1, \sigma = <\sigma_1, \ldots, \sigma_n>$, *entón* ω *é unha posición de* σ_i *e* $\sigma/\pi = \sigma_i/\omega$.

A substitución que se obtén de reemplazar a substitución σ/π por θ represéntase por $\sigma[\pi \leftarrow \theta]$. No caso de que non coincidan os codominios de σ e θ, é preciso compoñer antes as substitucións coas respectivas inclusións $Cod(\sigma) \cup Cod(\theta)$. Se $\sigma/\pi : \{i\} \rightarrow X$, por abuso da lingoaxe, poderá confundirse con $(i)(\sigma/\pi)$.

Exemplo 2.13 *Sexa $\sigma : \{1,2\} \rightarrow \{x\}$ dada por : $1\sigma = f(gx, hx)$, $2\sigma = kx$, entón*

- $1(\sigma/1) = f(gx, hx)$, $1(\sigma/2) = kx$,
- $1(\sigma/(1 \cdot 1)) = g(x)$, $2(\sigma/(1 \cdot 1)) = h(x)$,
- $1(\sigma/(1 \cdot 1 \cdot 1)) = x = 1(\sigma/1 \cdot 1 \cdot 2)$

2.3.4 Igualdade, 2-Categorías e Sesqui-Categorías

0,1,2-Células e 2-Categorías

Dado que as relacións de reescrita empregan unha operación ó composición, tense empregado a noción de 2-categoría para incorporar a reescrita a partir de 2-células entre morfismos (e.g. Power e Meseguer[142, 123]). Rydeheard e Stell [162], mostraron que a categoría de substitucións pódese dotar dunha estructura de 2-categoría empregando como 2-células as derivaciones entre substitucións que as fan iguais ecuacionalmente.

Definición 2.44 *Sexa $(A, \circ, 1)$ unha categoría chamada categoría base, cuios obxectos serán chamados 0-células e cuios morfismos se denominarán 1-células. Dise que A forma unha* 2-categoría *se se verifican as seguintes condicións :*

1) *para cada par de morfismos f e g de igual dominio X e codominio Y, existe un conxunto $AA[f,g]$ de 2-células, expresadas como $X \overset{f}{\underset{g}{\Downarrow \alpha}} Y$, cunha* composición vertical $\cdot$ *e unha identidade id_f para toda 1-célula f tal que $(AA[f,g], \cdot, id)$ é unha categoría. O inicio e destino das 2-células de $AA[f,g]$ son respectivamente f e g,*

2) *para todo par de 2-células : $X \overset{f}{\underset{g}{\Downarrow \alpha}} Y$ e $Y \overset{f'}{\underset{g'}{\Downarrow \beta}} Z$ existe unha* composición horizontal $X \overset{f' \circ f}{\underset{g' \circ g}{\Downarrow \alpha * \beta}} Z$ *coa propiedade asociativa,*

3) *se se teñen as seguintes células : $X \overset{f}{\underset{h}{\overset{g}{\Downarrow \alpha}{\Downarrow \alpha'}}} Y \overset{f'}{\underset{h'}{\overset{g'}{\Downarrow \beta}{\Downarrow \beta'}}} Z$, é necesario que se verifique a seguinte igualdade :*

$$(\alpha \cdot \alpha') * (\beta \cdot \beta') = (\alpha * \beta) \cdot (\alpha' * \beta') \quad \textit{(lei de intercambio)},$$

4) *se X, Y son 0-células, 1_X a 1-célula identidade en X e id_h, id_k, id_{1_X} as 2-células identidades de h,k e 1_X respectivamente, éstas son tamén as identidades para a composición horizontal, Por elo $id_h * id_k = id_{k \circ h}$ e se $X \overset{f}{\underset{g}{\Downarrow \alpha}} Y$ cúmplese que : $\alpha * id_{1_Y} = \alpha = id_{1_X} * \alpha$; i.e.*

$$\left(X \underset{}{\overset{}{\Downarrow \alpha}} Y\right) * \left(Y \underset{1_Y}{\overset{1_Y}{\Downarrow id}} Y\right) = X \underset{}{\overset{}{\Downarrow \alpha}} Y = \left(Y \underset{1_Y}{\overset{1_Y}{\Downarrow id}} Y\right) * \left(X \underset{}{\overset{}{\Downarrow \alpha}} Y\right)$$

A 2-célula $X \underset{g}{\overset{f}{\Downarrow \alpha}} Y$ denótase tamén como $\alpha : f \Rightarrow g$ cando non sexa necesario indicar os respectivos dominio e codominio de f e g. Dada unha 2-célula $X \underset{g}{\overset{f}{\Downarrow \alpha}} Y$, as 1-células f, g serán representadas como $s(\alpha)$ e por $t(\alpha)$ respectivamente para indicar a súa relación con α.

Sexan $f : X \to Y$, unha 1-célula, $Y \underset{h}{\overset{g}{\Downarrow \alpha}} Z$, $U \underset{l}{\overset{k}{\Downarrow \beta}} X$, 2-células, as composicións $X \underset{f}{\overset{f}{\Downarrow id_f}} Y \underset{h}{\overset{g}{\Downarrow \alpha}} Z$, $U \underset{l}{\overset{k}{\Downarrow \beta}} X \underset{f}{\overset{f}{\Downarrow id_f}} Y$ serán denotadas por $f * \alpha$ e por $\beta * f$. Se $\alpha : f \Rightarrow g, \beta : f' \Rightarrow g'$, pola lei de intercambio tense que :

$$\alpha * \beta = (f * \beta) \cdot (\alpha * g') \quad \underset{g'og}{\overset{f'of}{\Downarrow}} \begin{matrix} g'of \Downarrow f*\beta \\ \alpha * g' \end{matrix} = \underset{g'og}{\overset{f'of}{\Downarrow}} \begin{matrix} f'og \Downarrow \alpha * f' \\ g * \beta \end{matrix}$$

Se a categoría A ten coproductos, existe unha forma de descompoñer as 2-células en 2-células simples, o que se realiza compoñendo as inxeccións coas 2-células :

Sexa $X = \coprod_{i \in F} X_i$, con $j_i : X_i \to X$ a correspondente i-inxección, e $X \underset{g}{\overset{f}{\Downarrow \alpha}} Y$ unha 2-célula. Compoñendo horizontalmente as inxeccións con α obtense unha familia de 2-células $\{X_i \underset{goj_i}{\overset{foj_i}{\Downarrow \alpha_i}} Y\}$ onde $\alpha_i = j_i * \alpha$.

Unha propiedade de interese dentro dunha 2-Categoría é a existencia dunha forma de coproductos de 2-células.

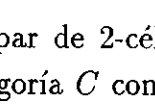
Definición 2.45 *Sexa A unha 2-categoría na que a categoría base ten coproductos. O coproducto de dous obxectos a, b é outro obxecto $a + b$ e un par de 1-células: $j_a \to a + b, j_b \to a + b$, tal que para calquera outro obxecto c, existe un isomorfismo entre as categorías $AA[a+b, c]$ y $AA[a, c] \times AA[b, c]$. Neste caso, dadas a, b, c 0-células, f, f', g, g' 1-células e $\alpha : f \Rightarrow g$, $\beta : f' \Rightarrow g$, tense unha equivalencia entre a 2-célula*

$$a + b \underset{<a,g>}{\overset{<f,f'>}{\Downarrow <\alpha,\beta>}} c, \quad \textit{e o par de 2-células} \left(a \underset{g}{\overset{f}{\Downarrow \alpha}} c, \quad b \underset{g'}{\overset{f'}{\Downarrow \beta}} c\right).$$

Deste xeito asegúrase que non só unha 2-célula pódese descompoñer noutras 2-células máis simples, senon tamén a existencia e unicidade dunha 2-célula α dada a partir das correspondentes α_i.

Se $\mathcal{T}eo$ é unha teoría ecuacional, a categoría de substitucións $Catsubst$ pódese dotar dunha estructura de 2-categoría, mediante a inclusión de 2-células nas seguintes condicións :

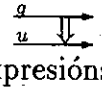
- para toda igualdade $s =_E t$, existe unha 2-célula $\{i\} \underset{\tau}{\overset{\sigma}{\Downarrow \alpha}} X$, onde $Var(s,t) = X$, $i\sigma = s$, $i\tau = t$,. Pola simetría de $=_E$ debe existir igualmente outra no senso inverso de τ a σ,
- se existen 2-células $\{i_j\} \underset{\tau_j}{\overset{\sigma_j}{\Downarrow \alpha_j}} X$ $1 \leq j \leq n$, entón existe $\coprod_{1 \leq j \leq n} \{i_j\} \underset{\tau}{\overset{\sigma}{\Downarrow \alpha}} X$, onde $\sigma = <\sigma_1, \ldots, \sigma_n>$, $\tau = <\tau_1, \ldots, \tau_n>$.

Dado un conxunto de ecuacións E, o problema de cómo obter a categoría $FinKleisli_E$ a partir da 2-categoría extensión de $Catsubst$, ten unha solución xeral en toda 2-categoría $\mathcal{A}$, establecendo en primeiro lugar unha relación entre as 1-células f, g dada por : $f \equiv g$ se existe un par de 2-células α, β, tales que  . Unha vez dada a relación $\equiv$, defínese a categoría C como aquela que ten como obxectos as 0-células de $\mathcal{A}$, e como morfismos as clases (módulo $\equiv$) de 1-células de $\mathcal{A}$.

Entre a categoría base A de $\mathcal{A}$ e C existe un funtor

$$A \xrightarrow{F_\equiv} C$$
$$F_\equiv(a) = a,\ F_\equiv(f : a \to b) = [f]_\equiv : a \to b.$$

Naturalmente, a categoría cociente C para unha teoría dada por E, é a categoría $FinKleisli_E$.

En certos casos, pode non ser necesario que exista unha 2-célula entre dúas 1-células consideradas equivalentes. Esto se verifica cando existe unha 2-categoría máis restrinxida (con menos 2-células), na que a propia relación de equivalencia $\equiv$ pode ser obtida por : $f \equiv g$ se existe unha 1-célula u e un par de 2-células  ,  . Con elo, evítase empregar 2-células mutuamente inversas para conectar dúas expresións igualadas. Esta situación é semellante ó reemplazamento da igualdade de termos pola reescrita ecuacional.

Comparando as relacións $=_E$ e as propiedades das 2-células, obsérvase que :

- a reflexividade de $=_E$ correspóndese coa existencia das 2-células id_f,
- a transitividade de $=_E$ é equiparable á composición vertical de 2-células,
- a composición horizontal permite expresar a monotonía e a estabilidade.

Para observar cómo a composición horizontal asegura a monotonía, sexan : f un operador n-ario, $\delta : \{i\} \to X = \{x_1, \ldots, x_n\}$, $i\delta = f(x_1, \ldots, x_n)$, e $\alpha = <\alpha_1, \ldots, \alpha_n>$ onde $\{x_i\}$  Y. Entón a composición $\delta * \alpha$ corresponde á aplicación das regras α_i no interior do termo $(x\delta) <\sigma_1, \ldots, \sigma_n>$.

Con esta mesma notación, a composición $\alpha_i * \lambda$, onde $Y \xrightarrow{\lambda} Z$, corresponde á estabilidade : a partir da igualdade $x_i\sigma =_E x_i\tau$ dada por α_i obtense a $x_i\sigma\lambda =_E x_i\tau\lambda$, dada por $\alpha_i\lambda$.

Pódense observar as seguintes diferencias entre as ecuacións e as 2-células en $Catsubst$:

- as 2-células atópanse etiquetadas. Obsérvese que no contexto clásico esta notación é, en moitos casos, engadida. E.g. $s =^1_E t\ (l \to r, \pi, \rho)$,
- a orientación das 2-células pode facer presupoñer que $\Rightarrow$ é unha relación de reescrita,

- a incorporación de coproductos de 2-células permite aplicar á vez varias regras en posicións independentes, pero sobre todo actuar sobre un elemento σ_i dun conxunto $\{\sigma_1, \ldots, \sigma_n\}$. Toda 2-célula da forma $I \overset{\sigma}{\underset{\theta}{\Downarrow \alpha}} X$ con $|I| > 1$ pódese descompoñer en 2-células $\{i\} \overset{\sigma_i}{\underset{\theta_i}{\Downarrow \alpha_i}} X$ compoñendo coas inclusións $j_i : \{i\} \rightarrow I$.

Presentacións en 2-Categorías

A continuación, móstrase como se pode obter unha 2-categoría a partir dunha categoría base A e dunha presentación R de 2-células básicas. Para elo, faise preciso incorporar tódalas 2-células que se poden xerar recursivamente a partir dos elementos de R, incluíndo:

- as identidades $id_f : f \Rightarrow f$ para todo morfismo $f : a \rightarrow b$ en A,
- as composicións verticais $\alpha \cdot \beta$ para todo par de 2-células componibles i.e. $s(\beta) = t(\alpha)$,
- a composición $\alpha * \beta$ se $cod(s(\alpha)) = dom(s(\beta))$.

Se a categoría ten coproductos, dadas as 2-células α_i, $1 \leq i \leq n$, incorpóranse asemade as 2-células :

- $I \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow \alpha}} X$, $\alpha = <\alpha_1, \ldots, \alpha_n>$, onde $I = \coprod_{1 \leq i \leq n} dom(s(\alpha_i))$, $s(\alpha) = <s(\alpha_1), \ldots, s(\alpha_n)>$, $t(\alpha) = <t(\alpha_1), \ldots, t(\alpha_n)>$.

  Para que se verifiquen os axiomas, compre identificar tódalas 2-células dadas por :

1. $\alpha : f \Rightarrow g$ con $id_f \cdot \alpha$ e con $\alpha \cdot id_g$. Polo tanto
   $\overset{f}{\underset{g}{\Downarrow \alpha}} = \overset{f}{\underset{g}{\Downarrow id_f}}, = \overset{\Downarrow \alpha}{\underset{\Downarrow id_g}{\longrightarrow}} = \overset{}{\underset{}{\Downarrow \alpha}}$,
2. α e as composicións horizontais $1_X * \alpha$ e $\alpha * 1_Y$. Así $X \overset{1_X}{\rightarrow} X \overset{}{\underset{}{\Downarrow \alpha}} Y = X \overset{}{\underset{}{\Downarrow \alpha}} Y \overset{1_Y}{\rightarrow} Y = X \overset{}{\underset{}{\Downarrow \alpha}} Y$,
3. $(\alpha \cdot \beta) \cdot \gamma$ e $\alpha \cdot (\beta \cdot \gamma)$ para as 2-células α, β, γ, se estas composicións se atopan definidas,
4. $(\alpha * \beta) * \gamma$ e $\alpha * (\beta * \gamma)$ se para éstas 2-células está definida a composición horizontal,
5. $(\alpha \cdot \alpha') * (\beta \cdot \beta') = (\alpha * \beta) \cdot (\alpha' * \beta')$ cando se atopan definidas as respectivas composicións.

Se a categoría base ten coproductos, para a existencia de coproductos de 2-células, é preciso facer coincidir as seguintes 2-células :

- $j_{X_i} * <\alpha_1, \ldots, \alpha_n>$ e α_i onde $X_i \overset{j_{X_i}}{\rightarrow} \coprod X_i \overset{}{\underset{}{\Downarrow <\alpha_1, \ldots, \alpha_n>}} Y$,

- $<\alpha, <\beta, \gamma>> = <<\alpha, \beta>, \gamma> = <\alpha, \beta, \gamma>$,
- $<\alpha_1, \ldots, \alpha_n> \cdot <\beta_1, \ldots, \beta_n>$ debe facerse idéntica a $<\alpha_1 \cdot \beta_1, \ldots, \alpha_n \cdot \beta_n>$ se $t(\alpha_i) = s(\beta_i)$, $1 \leq i \leq n$,
- $<\alpha_1, \ldots, \alpha_n> * \beta$ e $<\alpha_1 * \beta, \ldots, \alpha_n * \beta>$

Dada a signatura Σ, e un sistema de reescrita R, *Catsubst*, convírtese nunha 2-categoría con coproductos mediante a seguinte presentación : para toda $l = r \in R$, incorpóranse un par de 2-células : $\{x\} \overset{\sigma}{\underset{\theta}{\Downarrow \alpha}} Var(l, r)$, onde $l = x\sigma$, $r = x\theta$, e a súa simétrica $\theta \overset{\alpha^{-1}}{\Rightarrow} \sigma$.

Exemplo 2.14 *o axioma asociativo nunha teoría cunha operación binaria f vén expresado pola 2-célula : $\{i\} \overset{\sigma}{\underset{\theta}{\Downarrow \alpha}} \{x, y, z\}$ con $(i)\sigma = f(x, f(y, z)))$ e $(i)\theta = f(f(x, y), z))$.*

Stell [174] proba que a lei de intercambio, esixe a conmutatividade de certo tipo de derivacións *outermost* e *innermost*.

Definición 2.46 *Dado un sistema de reescrita R, unha derivación $t_0 \rightarrow_{\pi_0, l \rightarrow r, \rho} t_1 \rightarrow_{\pi_1, l' \rightarrow r', \rho'} t_2 \rightarrow_{\pi_2, l' \rightarrow r', \rho'} \ldots t_{n-1} \rightarrow_{\pi_{n-1}, l' \rightarrow r', \rho'} t_n$ é aniñada outermost equivalente á derivación aniñada innermost $t_0 \rightarrow_{\omega_0, l' \rightarrow r', \rho'} t_1' t_1 \rightarrow_{\omega_1, l' \rightarrow r', \rho'} t_2' \ldots t_{m-1} \rightarrow_{\pi, l \rightarrow r, \delta} t_m$ se existe $x \in Var(l)$ e posicións $\pi'', \pi_1', \ldots, \pi_n'$ tal que para todo i, $1 \leq i \leq n$, $\pi_i = \pi \cdot \pi_i' \cdot \pi''$ onde $\{\pi_i'\} = \Pi_x(r)$ e para todo j, $0 \leq j \leq m-1$, existen posicións ω_j', $\{\omega_j' | 0 \leq j \leq m-1\} = \Pi_x(l)$, $\omega_j = \pi_0 \cdot \omega' j \cdot \pi''$.*

O resultado de Stell, baséase en que pola lei de intercambio $(s(\alpha) * \beta) \cdot (\alpha * t(\beta)) = (\alpha * s(\beta)) \cdot (t(\alpha) * \beta)$ e se α e β son 2-células dadas directamente por $l \rightarrow r$ e $l' \rightarrow r'$ respectivamente, o primeiro membro da igualdade é a correspondente derivación aniñada innermost e aniñada outermost para a segunda.

Para que a 2-categoría xerada teña coproductos, compre igualar as diferentes secuencias de 2-células empregadas en posicións independentes. Esto é, sexan $I_1 \overset{\sigma_1}{\underset{\tau_1}{\Downarrow \alpha}} X$, $I_2 \overset{\sigma_2}{\underset{\tau_2}{\Downarrow \beta}} X$. As condicións do corproducto $< \alpha, \beta >$ son verificadas tanto por $<id_{\sigma_1}, \beta> \cdot <\alpha, id_{\tau_2}>$ como por $<\alpha, id_{\sigma_2}> \cdot <id_{\tau_1}, \beta>$, e por elo este par debe ser igualado (e unha 2-célula é polo tanto unha clase de equivalencia para estas relacións).

Incorporación de regras condicionais

Xa se mencionou o interese de empregar as regras condicionais para integrar a programación lóxica e funcional. Amais, existen teorías dadas por conxuntos cun número elevado (posiblemente infinito) de ecuacións que se poden obter a partir dunha ou varias ecuacións condicionais.

Probarase que o contexto de 2-categoría non é o apropiado para incluír as presentacións condicionais. Dada a igualdade condicional $s_1 \stackrel{?}{=} u_1, \ldots, s_n \stackrel{?}{=} u_n \Rightarrow l = r$, deséxase convertir esta expresión nunha 2-célula. En primeiro lugar deben explicitarse os respectivos conxuntos de variables. Como no caso non condicional, pode supoñerse que $X = Var(l, r)$. Sen embargo, no corpo da regla condicional pódense atopar variables extra, non presentes en X. Polo tanto, considérase que $Var(s_1, \ldots, s_n, u_1, \ldots, u_n) = X + X^e$. [12]

Por outra parte, para convertir a *Catsubst* nunha 2-categoría, as 2-células teñen necesariamente como *inicio* e *destino* as substitucións (de igual *dominio* e *imaxe*) igualadas pola regra condicional. Amais, as ecuacións sen condicións deben atoparse incluídas no caso condicional. Polo tanto, se α é unha 2-célula correspondente á ecuación condicional anterior, $s(\alpha) = \sigma$, $t(\alpha) = \tau$, $i\sigma = l$, $i\tau = r$. Será preciso incorporar unha serie de 2-células, de forma que, mediante unha nova relación $\equiv$, se obteña a equivalencia correspondente á igualdade de substitucións na teoría.

Definición 2.47 *Unha 2-célula condicional α é un par* $\left(I \xrightarrow[t(\alpha)]{s(\alpha)} X_\alpha,\ I^c_\alpha \xrightarrow{c_\alpha} X_\alpha + X^c_\alpha\right)$. *A 1-célula $s(\alpha)$ é chamada* inicio *de α e o* destino *de α é $t(\alpha)$. O conxunto de variables da 2-célula condicional é $X_\alpha + X^e_\alpha$.*

- *a 2-célula condicional identidade id_σ vén dada por* $(I \xrightarrow[\tau]{\sigma} X,\ \emptyset \longrightarrow X)$,
- *a composición vertical $\alpha \cdot \beta$ das 2-células* $\left(I \xrightarrow[t(\alpha)]{s(\alpha)} X,\ I^c_\alpha \xrightarrow{c_\alpha} X + X^e_\alpha\right)$ *e* $\left(I \xrightarrow[t(\beta)]{s(\beta)} X, I^c_\beta \xrightarrow{c_\beta} X + X^e_\beta\right)$ *obtense empregando as inxeccións*

$$j_{X^e_\alpha} : X + X^e_\alpha \rightarrowtail X + X^e_\alpha + X^e_\beta,$$
$$j_{X^e_\beta} : X + X^e_\beta \rightarrowtail X + X^e_\alpha + X^e_\beta$$

é a 2-célula condicional $\left(I \xrightarrow[t(\beta)]{s(\alpha)} X, I^c_\alpha + I^c_\beta \xrightarrow{c} X + X^e_\alpha + X^e_\beta\right)$ *onde* $c = < c_\alpha \circ j_{X^e_\alpha}, c_\beta \circ j_{X^e_\beta} >$, $c' = < c'_\alpha \circ j_{X^e_\alpha}, c'_\beta \circ j_{X^e_\beta} >$.

Con esta composición $Catsubst[X, Y]$ é unha categoría e inclúe o caso sen condicións, posto que :

- as identidades id_σ son neutras para esta composición,
- a composición é asociativa por selo a composición de morfismos e coproducto,
- unha 2-célula sen condicións toma a forma $(I \xrightarrow[\tau]{\sigma} X,\ \emptyset \longrightarrow X)$.

Acerca da posibilidade de introducir as 2-células condicionais como 2-células dunha 2-categoría, establécese a seguinte hipótese de traballo :

[12] a expresión X^e procede de *variables extra*.

Dado un conxunto de 2-células condicionais, existe unha composición horizontal $*$ que convirte a *Catsubst* nunha 2-categoría coas anteriores composicións vertical e identidade, e incorporando o significado habitual da igualdade condicional[13].

Para a composición horizontal de $\left(I \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow \alpha}} X, \quad I^c_\alpha \overset{c_\alpha}{\underset{c'_\alpha}{\rightarrow}} X + X^c_\alpha\right)$ con $\left(X \overset{s(\beta)}{\underset{t(\beta)}{\Downarrow \beta}} Y, I^c_\beta \overset{c_\beta}{\underset{c'_\beta}{\rightarrow}} Y + Y^c_\beta\right)$, precísase compoñer os respectivos inicio e destino, esto é, $s(\alpha * \beta) = s(\alpha)\, s(\beta)$, $t(\alpha * \beta) = t(\alpha)\, t(\beta)$.

As condicións de α deben compoñerse necesariamente con $s(\beta)$ ou con $t(\beta)$. Esto é necesario por dúas razóns diferentes. Dunha parte, para cumplir o requisito de que as variables do problema se atopen incluídas nas variables da parte condicional. Por outra parte, asemade é preciso para evitar regras incorrectas obtidas por instanción como se observa en :

Exemplo 2.15 *Sexan α, β correspondentes ás regras condicionais $x \overset{?}{=} c \overset{\alpha}{\Rightarrow} f(x) \rightarrow g(x)$, $\overset{\beta}{\Rightarrow} a \rightarrow b$. A composición horizontal $\alpha * \beta$ sen instanciación das condicións, dá orixe á nova regra $x \overset{?}{=} c \Rightarrow f(a) \rightarrow g(b)$. Obsérvese que o significado desta regra non se corresponde ó agardado : a regla iguala polo tanto a $f(a)$ e a $g(b)$, pero na teoría E dada por α e β, $f(a) \neq_E g(b)$. En cambio, ó facer a composición das condicións $x \overset{?}{=} c$ con $s(\beta)$ obtense á nova regra $a \overset{?}{=} c \Rightarrow f(a) \rightarrow g(b)$, que é correcta na teoría dada por α e β.*

Por elo, pode supoñerse que en $\alpha * \beta$, as condicións c_α, c'_α son compostas con $s(\beta)$.

Probarase que, con esa composición horizontal, non se verifica, en xeral, a lei de intercambio.

Sexan $(I \overset{}{\underset{}{\Downarrow \alpha}} X,\ I^c_\alpha \overset{c_\alpha}{\underset{c'_\alpha}{\rightarrow}} X + X^e_\alpha)$, $(I \overset{}{\underset{}{\Downarrow \alpha'}} X,\ I^c_{\alpha'} \overset{c_{\alpha'}}{\underset{c'_{\alpha'}}{\rightarrow}} X + X^e_{\alpha'})$ un par de 2-células compoñibles verticalmente e $(X \overset{}{\underset{}{\Downarrow \beta}} Y,\ I^c_\beta \overset{c_\beta}{\underset{c'_\beta}{\rightarrow}} X + X^e_\beta)$, $(I \overset{}{\underset{}{\Downarrow \beta'}} X,\ I^c_{\beta'} \overset{c_{\beta'}}{\underset{c'_{\beta'}}{\rightarrow}} X + X^e_{\beta'})$ outro par igualmente compoñibles. Para simplificar a notación, as correspondentes inxeccións non serán indicadas.

A primeira compoñente das condicións de $\alpha \cdot \alpha'$ é $< c_\alpha,\ c_{\alpha'} >$, mentras que a de $\beta \cdot \beta'$ é $< c_\beta,\ c_{\beta'} >$.

As condicións da composición horizontal $(\alpha \cdot \alpha') * (\beta \cdot \beta')$ obtéñense de xuntar as de $\alpha \cdot \alpha'$ (instanciadas por $s(\beta \cdot \beta')$) coas de $\beta \cdot \beta'$. Polo tanto $c_{(\alpha \cdot \alpha') * (\beta \cdot \beta')} =$ $<< c_\alpha, c_{\alpha'} > s(\beta),\ < c_\beta, c_{\beta'} >>$. Aplicando a unicidade do coproducto, pódese expresar como $< c_\alpha\, s(\beta), c_{\alpha'}\, s(\beta), c_\beta, c_{\beta'} >$.

A primeira compoñente das condicións de $\alpha * \beta$ é $< c_\alpha\, s(\beta), c_\beta >$ e a de $\alpha' * \beta'$ é $< c_{\alpha'}\, s(\beta'), c_{\beta'} >$. Para a composición vertical $(\alpha * \beta) \cdot (\alpha' * \beta')$ a primeira compoñente da composición é : $< c_\alpha\, s(\beta), c_\beta, c_{\alpha'}\, s(\beta'), c_{\beta'} >$ que, polas propiedades do coproducto, pódese expresar como $< c_\alpha\, s(\beta), c_{\alpha'}\, s(\beta'), c_\beta, c'_\beta >$.

[13]unha definición de éste significado será dada posteriormente.

Obsérvase que a condición correspondente a $I^c_{\alpha'}$ é, no primeiro caso, $c_{\alpha'}\, s(\beta)$ e no segundo $c_{\alpha'}\, s(\beta')$. Aplicando un razonamento semellante e, por simetría, obtense un resultado análogo para a outra instanciación posible empregada na composición horizontal $\Box$.

Esto proba a falsedade da conxetura e a necesidade de empregar unha estructura diferente da 2-categoría para a igualdade condicional.

Stell [174], observou que a composición horizontal nunha 2-categoría non era estrictamente necesaria para a representación da monotonía e estabilidade da igualdade ecuacional e que se pode esixir simplemente a existencia dunha composición pola esquerda e outra pola dereita de substitucións con 2-células, aclarando el significado de las composiciones horizontales. Amais desta circunstancia, algunhas outras melloras, como a de incluír as categorías de morfismos como l-categorías no senso de Mitchell[127], e a de integrar a rescrita en paralelo de Huet [87], induciron a Stell a introducir a noción de sesqui-categoría. A continuación probase que a igualdade condicional sí que se pode incorporar á categoría de substitucións producindo unha sesqui-categoría.

Sesqui-Categorías

Definición 2.48 *Unha categoría* $(A, \circ, 1)$ *de 0-células (obxectos) e 1-células (morfismos) é una* sesqui-categoría *se se verifican as condicións:*

1. *Para todo par de morfismos* f, g *de igual dominio* X *e codominio* Y*, existe un conxunto de 2-células* $AA[f,g]$ *cunha composición vertical* $\cdot$ *e unha identidade* id *tal que* $(AA[f,g], \cdot, id)$ *é unha categoría.*

2. *Existe un par de operacións externas, chamadas composicións horizontal pola esquerda e pola dereita, (denotadas ambas como* $*$*), tales que para cada 2-célula* $X \underset{}{\overset{}{\Downarrow \alpha}} Y$ *e para as 1-células* $Z \xrightarrow{f} X$, $Y \xrightarrow{g} U$ *existe unha 2-célula* $f * \alpha$, $\alpha * g$, *que verifican :*

   - $1_X * \alpha = \alpha = \alpha * 1_Y$, $h * id_k = id_h * k = id_{k \circ h}$,
   - *se* f, g, h, k *son 1-células e* α, β *son 2-células para as que as operacións están ben definidas, tense que* $(g \circ f) * \alpha = f * (g * \alpha)$, $(\alpha * h) * k = \alpha * (k \circ h)$, $g*(\alpha*f) = (g*\alpha)*f$, $f*(\alpha\cdot\beta) = (f*\alpha)\cdot(f*\beta)$, $(\alpha\cdot\beta)*g = (\alpha*g)\cdot(\beta*g)$.

   A definición de coproducto nunha sesqui-categoría é idéntica á dada previamente para unha 2-categoría.

Dado un conxunto de 2-células básicas, éstas xeneran a sesqui-categoría incorporando as seguintes 2-células e igualdades:

1. a identidade id_f para toda 1-célula f,

2. a composición vertical $\alpha \cdot \beta$ para todo par de 2-células compoñibles xunto coas igualdades :

   - $id_f \cdot \alpha = \alpha \cdot id_g = \alpha$,
   - $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$

3. a composición horizontal pola esquerda $I \overset{f\,s(\alpha)}{\underset{f\,t(\alpha)}{\Downarrow f*\alpha}} Y$ de $I \xrightarrow{f} X$ con $X \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow \alpha}} Y$ e análogamente pola dereita $\alpha * g$ onde $Y \xrightarrow{g} Z$. Estas composicións verifican:

   - $(h \circ f) * \alpha = f * (h * \alpha)$,
   - $\alpha * (k \circ g) = (\alpha * g) * k$,
   - $1_X * \alpha = \alpha = \alpha * 1_Y$

Para a existencia de coproductos, amais precísase que :

4. dada unha familia $I_j \overset{}{\underset{}{\Downarrow \alpha_j}} X$ de 2-células, existe unha 2-célula $I \overset{}{\underset{}{\Downarrow \alpha}} X$, $\alpha = <\alpha_1, \ldots, \alpha_n>$, $I = \coprod_{1 \leq j \leq n} I_j$, cumplindo as propiedades:

   - $s(\alpha) = <s(\alpha_1), \ldots, s(\alpha_n)>$,
   - $t(\alpha) = <t(\alpha_1), \ldots, t(\alpha_n)>$,
   - $<\alpha, <\beta, \gamma>> = \; <<\alpha, \beta>, \gamma> \; = <\alpha, \beta, \gamma>$,
   - $j_{I_j} * \alpha = \alpha_j$,
   - $<\alpha_1, \ldots, \alpha_n> * g = <\alpha_1 * g, \ldots, \alpha_n * g>$,
   - $<\alpha_1 \ldots, \alpha_n> \cdot <\beta_1, \ldots, \beta_n> = <\alpha_1 \cdot \beta_1, \ldots, \alpha_n \cdot \beta_n>$ si $t(\alpha_j) = s(\beta_j)$, $1 \leq j \leq n$.

Para convertir a *Catsubst* nunha sesqui-categoría onde as 2-células son igualdades condicionais, coa composición vertical e identidades definidas previamente, é preciso dar as súas composicións pola dereita e pola esquerda de 2-células con substitucións. Sexa $(X \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow \alpha}} Y,\ I_\alpha \overset{c_\alpha}{\underset{c'_\alpha}{\to}} Y + Y^e_\alpha)$ unha 2-célula e $\sigma : I \to X$ unha substitución.

$$\sigma * \alpha = \Big(I \overset{\sigma\, s(\alpha)}{\underset{\sigma\, t(\alpha)}{\Downarrow \sigma*\alpha}} Y,\ I_\alpha \overset{c_\alpha}{\underset{c'_\alpha}{\to}} Y + Y^e_\alpha\Big).$$

A composición pola dereita $\alpha * \theta$ onde $\theta : Y \to Z$ vén dada por

$$\Big(X \overset{s(\alpha)\,\theta}{\underset{t(\alpha)\,\theta}{\Downarrow \alpha*\theta}} Z,\ I_\alpha \overset{c}{\underset{c'}{\to}} Z + Y^e_\alpha\Big), \quad c = c_\alpha \; <\theta, j_{X^e_\alpha}>,\ c' = c'_\alpha \; <\theta, j_{X^e_\alpha}> .$$

Obsérvese que a composición pola esquerda non modifica as condicións, mentras que sí que o fai a composición pola dereita. Elo débese a que esta última composición modifica realmente as variables do problema, o que non sucede coa outra, dada a diferencia entre a composición pola esquerda e pola dereita de 2-células con morfismos.

Como no caso da 2-categoría xerada por unha relación de reescrita, a existencia de coproductos obliga a identificar $\alpha \cdot \beta$ con $\beta \cdot \alpha$ se as posicións onde ten lugar a reescrita correspondente a α e a β son independentes. Esto xera unha relación de equivalencia e as 2-células da sesqui-categoría *Catsubst* son as clases para esa relación. Esta relación é compatible coa composición de 2-células con substitucións, dado que se α e β son 2-células aplicadas en posicións π, π' independentes, $\alpha * \sigma$ e $\beta * \sigma$ corresponden a aplicación da reescrita nesas mesmas posicións π, π' e $\{i\} \overrightarrow{\Downarrow} \alpha * \sigma Y$, $\{i\} \overrightarrow{\Downarrow} \beta * \sigma Y$ equivale a aplicar as mesmas reglas en posicións $\omega \cdot \pi, \omega \cdot \pi'$, onde $i\sigma/\omega = x$ que son novamente independentes (o caso $I \neq \{i\}$, é análogo).

Sexa Δ unha secuencia de pasos de reescrita e $\overline{\Delta}$ a clase de equivalencia (2-célula correspondente en *Catsubst*). Dado que todas as etapas teñen o mesmo número de pasos de reescrita, existe unha noción de lonxitude $|\overline{\Delta}|$ coerente coas composicións horizontais (pola esquerda e pola dereita) e co coproducto.

A lonxitude dunha 2-célula α nunha sesqui-categoría é o maior número natural n tal que α pode expresarse como composición de n 2-células que non son identidades, pero non se pode expresar empregando máis de n. Obsérvese que non precisa a existencia dunha lonxitude nas 1-células da categoría base.

2.53 *As 2-células xeradas a partir dun conxunto de 2-células condicionais e coas operacións e equivalencia mencionadas, convirten á categoría Catsubst nunha sesqui-categoría con coproductos.*

As condicións non interfiren na equivalencia, e as propiedades son consecuencia consecuencia directa da asociatividade e identidades para as substitucións.

Definición 2.49 *Sexa SC unha sesqui-categoría obtida de Catsubst incorporando un conxunto R de 2-células e sexa α unha 2-célula. A 2-célula β é unha* extensión *de α se*

- $\beta = \alpha$, *ou*
- *existe unha substitución σ, tal que $\beta = ex(\alpha) * \sigma$, ou ben σ é épica e $\beta = \sigma * ex(\alpha)$, onde $ex(\alpha)$ é unha extensión de α, ou*
- *existe unha extensión de α, $(ex(\alpha))$ e substitucións σ, τ tales que $\beta = <\sigma, ex(\alpha), \tau>$*

Dada a 2-célula α, dirase que $ex(\alpha)$ é unha extensión resultado de aplicar α en π, se existe unha substitución σ, tal que $s(ex(\alpha))/\pi = s(\alpha)\sigma$, e $t(ex(\alpha))/\pi = t(\alpha)\sigma$.

Defínese a lonxitude *de α por :*

- *se $\alpha = id_\sigma$, entón $lonxitude(\alpha) = 0$,*
- *se $\alpha \in R$, $lonxitude(\alpha) = 1$,*
- *se $\alpha = \beta \cdot \gamma$ entón $lonxitude\ (\alpha) = lonxitude(\beta) + lonxitude(\gamma)$,*

- *se* $\alpha = \beta * \sigma$, , *lonxitude*$(\alpha) =$ *lonxitude*(β),
- *se* $\alpha = <\alpha_1, \ldots, \alpha_n>$, *lonxitude*$(\alpha) = \Sigma_{1 \leq i \leq n}$*lonxitude*$(\alpha_i)$,
- *noutro caso pódese supoñer que* $\alpha = \sigma * \beta$ *onde* $\{i\} \xrightarrow{\sigma} X$. *Se* σ_{ep} *é isomorfismo entón* $\sigma = j_{\{i\}}$. *Noutro caso sexa* $\bar{\sigma}, \underline{\sigma}$ *a factorización do primeiro operador de* σ *e* *lonxitude*$(\alpha) =$ *lonxitude*$(\underline{\sigma} * \beta)$.

En forma semellante á igualdade condicional, a sesqui-categoría C xerada dende $Catsubst$ a partir dun conxunto de 2-células condicionais da lugar a novas sesqui-categorías.

Constrúese unha sucesión de sesqui-categorías A_i cuia base é $Catsubst$ e cuias 2-células veñen dadas por :

- A_0 ten como 2-células a :

  $I \underset{t(\alpha)\circ j_{X\circ\sigma}}{\overset{s(\alpha)\circ j_{X\circ\sigma}}{\Downarrow \alpha\sigma}} Y$ se existe unha 2-célula $(I \overset{\Downarrow \alpha}{\longrightarrow} X,\ I_\alpha^c \overset{c_\alpha}{\underset{c'_\alpha}{\rightrightarrows}} X + X_\alpha)$ de C e se verifica que $c_\alpha\sigma = c'_\alpha\sigma$ para unha substitución $\sigma : X \to Y$.

- A_i ten como 2-células a tódalas de A_{i-1} así como ás seguintes :

  $I \underset{t(\alpha)\circ j_{X\circ\sigma}}{\overset{s(\alpha)\circ j_{X\circ\sigma}}{\Downarrow \alpha\sigma}} Y$ se existen : unha 2-célula condicional $\left(I \overset{\Downarrow \alpha}{\longrightarrow} X,\ I_\alpha^c \overset{c_\alpha}{\underset{c'_\alpha}{\rightrightarrows}} X + X_\alpha\right)$ e unha 2-célula $\left(I_\alpha^c \overset{c_\alpha\sigma}{\underset{c'_\alpha\sigma}{\Downarrow}} Y, \emptyset \longrightarrow Y\right)$ de A_{i-1}.

Obtense unha nova sesqui-categoría A, que ten como 2-células a todas aquelas que son 2-células dalgunha A_i. A *profundidade dunha 2-célula* α de A é o menor i, tal que α é unha 2-célula de A_i. Obsérvese que a profundidade e lonxitude dunha 2-célula son independentes da existencia doutra 2-célula que conecte as mesmas substitucións.

2.54 *As composicións vertical e horizontal definidas, non modifican o significado da igualdade : dúas substitucións* $X \overset{\sigma}{\underset{\tau}{\rightrightarrows}} Y$ *están conectadas por algunha 2-célula sse son iguais na correspondente teoría ecuacional.*

Sexa E_i a igualdade condicional correspondente a tódalas derivacións de profundidade i. Por inducción nesta, probase que $\sigma \Rightarrow_{A_i} \tau$ sse para todo $x_i \in Dom(\sigma)$, $x_i\sigma =_{E_i} x_i\tau$. Xa que as ecuacións da teoría veñen dadas por $\cup_{i \geq 0} E_i$ e A contén a tódalas 2-células de cada A_i, obtense o resultado $\square$.

Definición 2.50 *Dadas as 2-células* $\alpha \in R$ *e* β, $cod(s(\alpha)) = X_\alpha$, $cod(s(\beta)) = X_\beta$, *presentan un par crítico se existe* $X_\alpha + X_\beta \xrightarrow{\sigma} Z$, *coigualador de* $s(\alpha)j_{X_\alpha}, s(\beta)j_{X_\beta}$. *O par crítico é* (γ, γ^{-1}), *onde* $\gamma = \left(I \underset{t(\beta)\circ\sigma}{\overset{t(\alpha)\circ\sigma}{\Downarrow \gamma}} Z,\ I_\alpha^c + I_\beta^c \overset{c}{\rightrightarrows} Z + X_\alpha^e + X_\beta^e\right)$ *e cuias condicións son* $c = <c_1, c_2>$, $c' = <c'_1, c'_2>$ *dadas por :*

$$c_1 : I^c_\alpha \rightarrow X_\alpha + X^e_\alpha \rightarrow X_\alpha + X_\beta + X^e_\alpha \rightarrow Z + X^e_\alpha \rightarrow Z + X^e_\alpha + X^e_\beta,$$
$$c_1 = c_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ <\sigma, 1_{X^e_\alpha}> \circ j_{Z+X^e_\alpha},$$

$$c_2 : I^c_\beta \rightarrow X_\beta + X^e_\beta \rightarrow X_\alpha + X_\beta + X^e_\beta \rightarrow Z + X^e_\beta \rightarrow Z + X^e_\alpha + X^e_\beta,$$
$$c_2 = c_\beta \circ j_{X_\beta + X^e_\beta} \circ <\sigma, 1_{X^e_\beta}> \circ j_{Z+X^e_\beta},$$

$$c'_1 : I^c_\alpha \rightarrow X_\alpha + X^e_\alpha \rightarrow X_\alpha + X_\beta + X^e_\alpha \rightarrow Z + X^e_\alpha \rightarrow Z + X^e_\alpha + X^e_\beta,$$
$$c'_1 = c'_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ <\sigma, 1_{X^e_\alpha}> \circ j_{Z+X^e_\alpha},$$

$$c'_2 : I^c_\beta \rightarrow X_\beta + X^e_\beta \rightarrow X_\alpha + X_\beta + X^e_\beta \rightarrow Z + X^e_\beta \rightarrow Z + X^e_\alpha + X^e_\beta,$$
$$c'_2 = c'_\beta \circ j_{X_\beta + X^e_\beta} \circ <\sigma, 1_{X^e_\beta}> \circ j_{Z+X^e_\beta}.$$

A 2-célula γ^{-1} só varía respecto de γ en que teñen intercambiados os respectivos inicio e destino.

Se β é unha extensión dunha 2-célula δ do conxunto R, tal que $s(\beta)/\pi = \delta$, entón o par crítico (γ, γ^{-1}) dise asemade que se forma dende a posición π de α coa 2-célula δ.

Dadas as 2-células α, β presentan un par crítico γ dado por $I \overset{s(\gamma)}{\underset{t(\gamma)}{\Downarrow \gamma}} Z$, $I^c_\alpha + I^c_\beta \overset{c_\gamma}{\underset{c'_\gamma}{\rightarrow}} Z + X^e_\gamma$ se se verifican :

- $\sigma = mgu(c_\alpha \circ j_{X_\alpha + X^e_\alpha}, s(\beta) \circ j_{X_\beta + X^e_\beta})$. *Sexa $\sigma'_{ep} \circ j_Z$ a factorización minimal de $j_{X_\alpha} \circ \sigma$,*
- $s(\gamma) = s(\alpha)\sigma'_{ep}$, $t(\gamma) = t(\alpha)\sigma'_{ep}$,
- $c_\gamma = <c_1, c_2>$, $c'_\gamma = <c'_1, c'_2>$ *onde*
  - $c_1 = c_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ \sigma : I^c_\alpha \rightarrow X_\alpha + X^e_\alpha \rightarrow Z + X^e_\gamma$
  - $c'_1 = c'_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ \sigma : I^c_\alpha \rightarrow X_\alpha + X^e_\alpha \rightarrow Z + X^e_\gamma$
  - $c_2 = c_\beta \circ j_{X_\beta + X^e_\beta} \circ \sigma : I^c_\beta \rightarrow X_\beta + X^e_\beta \rightarrow Z + X^e_\gamma$
  - $c'_2 = c'_\beta \circ j_{X_\beta + X^e_\beta} \circ \sigma : I^c_\beta \rightarrow X_\beta + X^e_\beta \rightarrow Z + X^e_\gamma$

Como no caso anterior, β pode tratarse da extensión dunha 2-célula δ.

Stell[174] proba que nunha sesqui-categoría existe unha noción próxima ós pares críticos : os *critical spans* que son pares de 2-células da forma $\overset{\Uparrow \beta}{\underset{\Downarrow \alpha}{\rightarrow}}$ que non teñen sido obtidos a partir de outros pares de 2-células, composicións e coproductos. Proba igualmente que os pares críticos en *Catsubst* forman un tipo concreto de *critical spans*, mostrando a capacidade da sesqui-categoría para incorporar un bó número de características da reescrita e igualdade ecuacional.

Xa que este traballo se orienta á unificación ecuacional, dadas as substitucións $I \xrightarrow{\sigma} X_\sigma$, e $I \xrightarrow{\theta} X_\theta$ se $X_\sigma \neq X_\theta$ non é posible a existencia de 2-células entre elas. Sen

embargo, pode suceder que se teñan inxeccións $X_\sigma \overset{j_{X_\sigma}}{\rightarrow} Z, X_\theta \overset{j_{X_\theta}}{\rightarrow} Z$, para as que existe unha 2-célula entre $\sigma \circ j_{X_\sigma}$ e $\theta \circ j_{X_\theta}$, en cuio caso, por abuso da lingoaxe, dirase que existe unha 2-célula entre σ e θ.

Dada unha sesqui-categoría dise que está dirixida por unha relación $\succ$ se as 2-células R xeradas pola sesqui-categoría están dirixidas por unha relación $\succ$. Pola semellanza coa igualdade de termos ecuacionais, as propiedades acerca da existencia e confluencia dun conxunto que permite dirixir as ecuacións, poden ser igualmente utilizadas neste novo contexto.

Capítulo 3

Unificación de Termos Libres

A unificación de termos libres, esto é, nunha teoría sen ecuacións, chámase tamén unificación sintáctica, e ten unha gran importancia no desenrolo dos algoritmos de unificación ecuacional. Neste capítulo realízase un repaso xeral dos algoritmos de unificación, incluíndo a técnica de factorización do unificador, así como unha comparación entre eles, indicando algunhas técnicas de acelerar a a unificación. Introdúcese unha forma de realizar a unificación de substitucións usando propiedades de categorías e próbase que a relación da idempotencia cos unificadores máis xerais, vén caracterizado na unificación de substitucións pola existencia dunha adxunción entre categorías de coigualadores.

3.1 Algoritmos de Unificación

3.1.1 Historia da unificación

Os inicios da teoría da unificación atópanse en E. Post (diario e notas), onde xa se indica a vantaxe de obter un representante xeral frente a tódalas posibles instanciacións dos termos, e en J. Herbrand[80] o cal, interesado na resolución de ecuacións, deu un algoritmo non determinístico para obter o unificador de dous termos; sen embargo, este algoritmo non foi inmediatamente aproveitado, a diferencia de outros resultados de Herbrand.

A finais dos anos cincuenta S. Kanger redescubre a unificación aínda que sen superar o desenrolo de Herbrand. En Novembro de 1962 M. Davis implementa un algoritmo de unificación nun IBM 7090 no Bell Telephone Laboratories baseado nunha combinación do procedemento de Davis-Putnam e en propostas realizadas por D. Prawitz[143], o cal dera unha especie de algoritmo de unificación pero sen permitir calesquera símbolos.

Tanto para o desenrolo da unificación como para o da proba automática, resulta de maior transcendencia o traballo de J.A. Robinson[150]. Este acuña os nomes de unificador e de unificador máis xeral, proba a existencia de este e utilízao para crear unha nova forma de deducción mediante a **Regra de Resolución** no que se

considera o primeiro sistema efectivo de mecanización da lóxica de primeiro orden. E menos coñecido o feito de que Mc Ilroy xa descubrira e utilizara en 1962 un algoritmo semellante ó de Robinson (como recoge Davis[48]).

A súa vez, nos traballos de J. R. Guard [79] suxeríronse outras extensións como a unificación de alto orden e a unificación ecuacional, desenroladas posteriormente.

Knuth e Bendix [108] deron un novo algoritmo de unificación e probaron que ésta é unha operación básica da reescrita de termos.

A unificación ten sido empregada tanto en :

- teorías ecuacionais [139],
- termos de Orden Superior (*Higher Order*) [85],
- termos de Xénero Ordenado ((*Order Sorted*)[124].
- lingoaxes de programación lóxica [42],
- ψunificación, CIL-unificación e operacións nas *feature structures* [171].

A unificación é tamén unha pieza fundamental na integración da programación lóxica e funcional, proba automática, inferencia de tipos na programación funcional, λ-cálculo, reescrita, verificación de hardware e sistemas de parsing orientados ó procesamento da lingoaxe natural [93, 87, 94, 121, 107] etc.

Xeralmente precísase dispor dun mecanismo de unificación sintáctico básico o máis potente e eficaz posible.

Os algoritmos de unificación sintácticos clasifícanse aquí en :

- algoritmos de unificación representando termos como palabras,
- algoritmos que utilizan grafos e outras estructuras para a representación,
- algoritmos que realizan a transformación de ecuacións,
- algoritmos de unificación paralela.

Algoritmos de unificación representando termos como palabras

Nos primeiros algoritmos, os datos eran gardados sinxelamente como cadeas de caracteres. Entre eles, dada a súa importancia histórica, destacan o algoritmo de Robinson e o de Knuth-Bendix.

Unha característica deste tipo de algoritmos está na simplicidade dos datos, xa que utilizan a expresión clásica dos termos a partir de símbolos funcionais, variables, comas e parénteses[1]. Non requiren punteiros, nen conversión a outra estructura e a sustitución resultante está dispoñible directamente.

[1] é coñecido que se os operadores non son multiádicos as comas e paréntese non son necesarios.

O conxunto de desacordo de A (A') obtense localizando a primeira posición na que non tódalas expresións de A teñen o mesmo símbolo e extraendo estes símbolos. As posicións recórrense de esquerda a dereita e utilízase un orden total nos símbolos de A no que tódalas variables son inferiores ós operadores.

Algoritmo 3.1 ([150]) *Unificación de Robinson :*

- *Entrada : Un conxunto A de expresións ben formadas.*
- *Saída : Un unificador idempotente de A, σ_A.*

Paso 1 *Facer* $k = 0, \sigma_0 = \varepsilon$ *e Seguir*
Paso 2 *Se* $A\sigma_k$ *non e unitario Enton Seguir*
Noutro caso $\sigma_A = \sigma_k$ *e Rematar.*
Paso 3 *Sexan* v_k, t_k *neste orden os dous primeiros elementos do conxunto* B_k *de desacordo de* $A\sigma_k$
Se v_k *e variable e* $v_k \notin Var(t_k)$ *Enton*
Facer $\sigma_{k+1} = \sigma_k \circ \{v_k/t_k\}$, $k = k + 1$ *e ir a Paso 2*
Noutro caso Rematar con Fallo

Robinson define o unificador máis xeral como a sustitución obtida no *Paso 2*, definición pouco formal que foi pulida por Chang e Lee [38]. Estes autores definen o *mgu* na sua forma actual[2].

Este algoritmo realiza a unificación dun conxunto de expresións e para elo, unifica pares de termos, pero ésto non se fai explícito a diferencia del método de transformación de ecuacións. Non é moi eficaz xa que :

- no *Paso 2* precísase :
  - aplicar a sustitución σ_k a toda a expresión, o que significa recorrela completamente para reemplazar a variable v_k,
  - calcular o conxunto de desacordo da expresión, o que obriga a comparar o termo t_k cos termos situados na mesma posición que íl.
- no *Paso 3*
  - deben buscarse os termos minimais, no orden lexicográfico, do conxunto de desacordo, o que, no peor caso, esixe comparar novamente entre sí tódolos termos correspondentes excepto a variable eliminada,
  - no caso de non existencia de "clash" (operadores distintos), faise preciso recorrer o termo t_k para comprobar que $v_k \notin Var(t_k)$.

[2]se ben teñen aparecido outras definicións semellantes[152] .

como consecuencia de elo pode tardarse moito en detectar os conflictos. Por eso este algoritmo resulta máis eficaz se o número de éxitos é alto, sendo un algoritmo con alto custe tanto en tempo como en espacio nos peores casos.

3.1 *Sexa A un conxunto finito non vacío de expresións ben formadas. Se A é unificable, ten un unificador máis xeral σ. Se θ é un unificador de A, entón $\sigma \leq \theta$. Se $A = \{s,t\}$ é unificable, existe un supremo de s e t en $\mathcal{T}_\Sigma/\equiv$.*

Proba. O algoritmo anterior debe rematar, xa que en cada iteración o número de variables de $A\sigma_k$ redúcese nunha unidade. Se o algoritmo remata no *Paso 2*, a substitución resultante é efectivamente un unificador pois o desacordo de $A\sigma_k$ é unitario. Probase que $\sigma \leq \theta$, aplicando inducción en k. Se $k = 0, \theta = \sigma_k\lambda_k$ por definición de σ_0. Suposto para k. Xa que σ_k ten sido calculado (*Paso 3*), transfírese o control ó *Paso 2*. Se $A\sigma_k$ é unitario, σ_k é unificador e por hipótese verifícase que $\sigma \leq \theta$.

Noutro caso é aplicado novamente o *Paso 3*. Pola minimalidade das variables, v_k é variable xa que noutro caso non sería unificable. Supoñendo que $v_k \in Var(t_k)$ e posto que $v_k\lambda_k = t_k\lambda_k$ e, dada a diferente lonxitude de ambos termos, chégase a unha contradicción. Así pois, non se detén no *Paso 3* xa que nese caso non sería unificable. Polo tanto obtense $\sigma_{k+1} = \sigma_k \circ \{v_k/t_k\}$.
Sexa $\lambda_{k+1} = \lambda_k[Dom(\lambda_k \backslash \{v_k\})]$. Falla por probar que $\theta = \sigma_{k+1}\lambda_{k+1}$. Tense que $\lambda_k = \{x_k/t_k\lambda_k\} \cup \lambda_{k+1} = \{x_k/t_k\lambda_{k+1}\} \cup \lambda_{k+1} = \{x_k/t_k\} \circ \lambda_{k+1}$, de onde se deduce $\theta = \sigma_k\lambda_k = \sigma_{k+1}\lambda_{k+1}$. Se σ é un *mgu* de s e t, a clase en $\equiv$ de $s\sigma = t\sigma$ é o supremo das clases de s e t □.

O unificador máis xeral é claramente idempotente, xa que σ_i é idempotente e a composición éo por aplicación da propiedade *2.12*.

Knuth e Bendix [108] no seu destacado artículo sobre reescrita, proban a existencia dunha solución máis xeral de todo problema resoluble da forma $s \stackrel{?}{=} t$ con variables $v_1, \ldots v_n$. Aínda que o algoritmo non é explícito, sinalan que a proba equivale a un algoritmo recursivo.

Algoritmo 3.2 ([108]) *Algoritmo de unificación de Knuth e Bendix*
Entrada : O par de termos s e t a unificar
Saída : O unificador máis xeral σ

Unificar (s,t) =
  Caso 1 Se $s,t \in V$ Enton
    Se $s = t$ Enton $\sigma = \varepsilon$
    Noutro caso $\sigma = \{s/t\}$
  Caso 2 Se $s \in V, t \notin V$ Enton
    Se $s \in Var(t)$ Enton Fallo e Parar
    Noutro caso $\sigma = \{s/t\}$

Caso 3 Se $t \in V, s \notin V$ *semellante o Caso 2*
Caso 4 Sexan $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_m)$ *Enton*
Se $f \neq g$ *ou ben* $n \neq m$ *Enton Fallo e Parar.*
Noutro caso $\Delta = \{(s_1, t_1), \ldots, (s_n, t_n)\}$
Se $n = 0$ *Enton* $\sigma = \varepsilon$
Noutro caso $r = 0, \sigma_0 = \varepsilon$
Mentras $r \leq n - 1$
Sexa θ *o resultado de Unificar* $(s_{r+1}\sigma_r, t_{r+1}\sigma_r)$.
Facer $\{ \sigma_{r+1} = \sigma_r \circ \theta,\ r = r + 1 \}$.

No que sigue, o paso da forma $s \in Var(t)$ *entón Fallo e parar* será chamado *test de ciclos*, e o da forma $f \neq g \ldots$, *chequeo de operadores*, ou *test de clashing*. Non hai sustanciais diferencias entre ambos algoritmos : se no de Robinson unifícanse dous ou máis termos (o que obriga a escoller os dous de menor orden), Knuth e Bendix fanno soamente para un par e, localmente para conxuntos de pares, estando no algoritmo destes máis claro o chequeo dos operadores xa que no de Robinson atópase oscurecido pola construcción do conxunto de desacordo e pola extracción dos elementos minimais. En calquera caso, os pares son leídos de esquerda a dereita e a sustitución obtida en cada paso compónse coa resultante do paso anterior.

Unha implementación do algoritmo de Knuth-Bendix na lingoaxe ML aparece no traballo de Fages[61]. Nesta implementación, a unificación de conxuntos de pares a partir de *unify*, realízase mediante a función local *unify_list* que se atopa definida no interior da propia *unify*.

Todos estes algoritmos teñen un alto custe en tempo e espacio, especialmente na operación de eliminación de variables xa que :

i) se precisa recorrer ambos pares de termos de cada vez que se elimina unha variable,

ii) o termo no que se instancia pode ter unha lonxitude exponencial no tamaño da entrada.

O seguinte exemplo mostra que o custe do algoritmo de Robinson pode chegar a ser exponencial no tamaño de a entrada.

Exemplo 3.1 $s = f(x_n, \ldots, x_2, x_1)$, $t = f(g(x_{n-1}, x_{n-1}), \ldots, g(x_1, x_1), g(x, x))$ *cuio mgu é* $\sigma = \{x_1/g(x, x), x_2/g(g(x, x), g(x, x)) \ldots, x_n/g(g(\ldots), g(\ldots))\}$ *e o tamaño de* $x_i\sigma$ *é* $2^{i+1} - 1$ *con* $1 \leq i \leq n$. *Xa que o tamaño de* s, t *no exemplo é lineal en* n, *o custe de espacio e tempo é exponencial.*

O custe do algoritmo de Robinson para un par de termos con un número de variables V fixo de cardinal v, e de lonxitude da entrada n é, no seu peor caso, polinómico en n de orden v.

Algoritmos que utilizan outras representacións

O alto custe do algoritmo de Robinson, é debido principalmente á necesidade de reemplazar cada variable pola sua instanciación en tódalas súas aparicións, co conseguinte copiado de símbolos. Esto pódese resolver empregando estructuras para os datos nas que o reemplazamento signifique tan só creación de punteiros das variables as súas instanciacións.

Xa Robinson[151] suxire reducir este custe mediante o uso dunha representación dos termos por táboas. Boyer e Moore [29] deron unha representación compartida que reducía o custe en espacio, pero a complexidade no peor tempo seguía sendo exponencial. Melloras notables son as de L. D. Baxter , de M. Venturini-Zilli e de Huet [16, 180, 85], quen mostrou un algoritmo case lineal gracias a considerar a unificación como un caso particular de clausura de clases de termos, utilizando as operacións de UNION e FIND[1] de mantemento de clases de equivalencia. O custe en tempo do algoritmo de Huet é de $\mathcal{O}(n \cdot \alpha(n))$ onde $\alpha(n)$ é a inversa da función de Ackermann, función que crece de forma extraordinariamente lenta. Paterson e Wegman [137] deron un algoritmo que é lineal no tamaño dos termos aínda que non chega a ser práctico debido ó custe de encabezamento (preparación) e ó gran número de punteiros.

Existen outros algoritmos tamén de custe case lineal ou lineal (J.A. Robinson, D. Kapur, M.S. Krishnamoorthy e P. Narendran, Escalada e Ghallab, Ružička e Privara) [152, 99, 60, 158] baseados tamén en representacións reducidas e grafos.

O algoritmo de Huet proporciona como resultado un par $(bool, \sigma)$, onde a variable *bool* indica se é ou non unificable e de selo, σ é un unificador máis xeral.

Algoritmo 3.3 ([85]) *Algoritmo de Huet*
Entrada : O par de termos s e t a unificar (en forma de grafo).
Saída : O grafo co resultado da unificación. Se este non ten ciclos devolve $(true, \sigma)$, onde σ é a sustitución que leva toda variable x do grafo en $FIND(x)$.

Unificar (s,t) =
  Sexa pila_pares $= \{(s,t)\}$
  Para cada no w *en* (s,t)
    Facer { *clase*$(w) = w$ }
  Mentras pila_pares $\neq \emptyset$
    Facer {
      Sexa (x,y) = *Pop(pila_pares)*
      Sexa u = *FIND(x)*
      Sexa v = *FIND(y)*
      Se $u \neq v$ *Enton*
        Se u *e* v *non son variables e simbolo*$(u) \neq$ *simbolo*(v) *ou*
          numero_subnos$(u) \neq$ *numero_subnos*(v)
        Enton $(Falso, \epsilon)$
        Noutro caso

Sexa $w = UNION(u, v)$
Se $w = v$ *e* u *e variable Enton clase*$(u) = v$
Se $w = u$ *e* v *e variable Enton clase*$(v) = u$
Noutro caso $/*$ u *e* v *non son variables* $*/$
Sexan $(u_1, \ldots, u_n)$ *os subnos de* u
Sexan $(v_1, \ldots, v_n)$ *os subnos de* v
Para $i = 1$ *ata* n
Facer {
$push\ ((u_i, v_i), pila\_pares)$ } }

O algoritmo anterior non realiza o chequeo de ciclos e polo tanto, non detecta se a instanciación dunha variable contén como subtermo a esa mesma variable. Por eso denomínase algoritmo de unificación racional (admite termos racionais). Esto pode ser correxido engadindo ó final algún dos algoritmos lineais de clasificación topolóxica para detectar se o grafo resultante contén ciclos.

No seguinte exemplo que corresponde á figura *3.1* poden observarse o grafo inicial, as transformacións realizadas no grafo e o grafo resultante.

Exemplo 3.2 *Sexan* $s = f(x, g(x))$, $t = f(g(y), g(z))$. *Os sucesivos pasos mostrados en la figura 3.1 son :*

- *Inicio : UNION(nós superiores),*
- *Pasos 2,3 : transmisión da conexión cara abaixo : usando UNION e* $push(u_i, v_i)$,
- *Paso 4 : FIND(x) busca o representante f(y) da clase de x. Este representante, convírtese asemade no representante da variable z.*

  O unificador $\{x/f(y), z/f(y)\}$ *obtense a partir de* $FIND(x), FIND(z)$ *e obsérvase que o supremo de s e t está explícitamente xerado.*

E esencial acelerar a obtención do representante da clase (FIND) e a reunión de clases (UNION). Para esto último emprégase un peso que indica o número de membros de cada clase a mover (UNION_WITH_WEIGHT) de forma que o número de movementos sexa o menor posible.

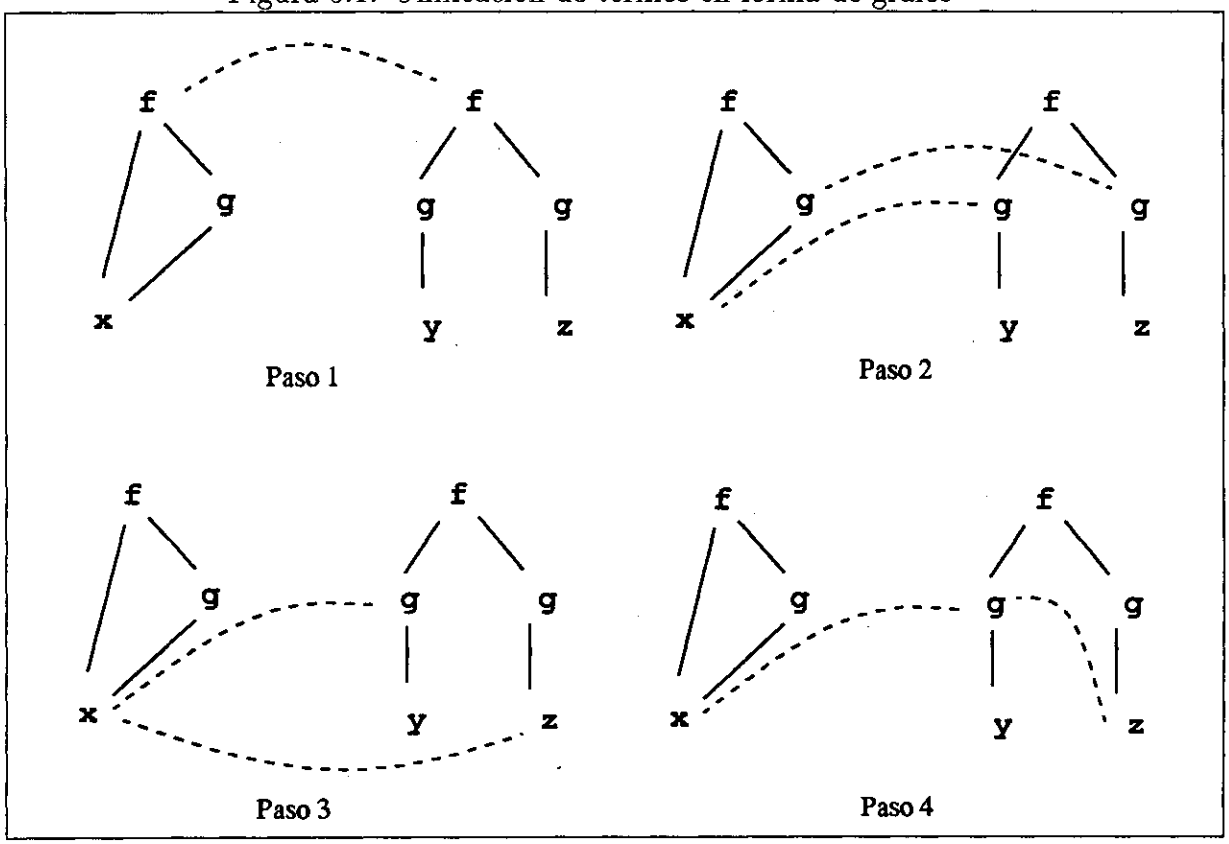
Cando se busca o representante dunha clase, colócanse novos punteiros dos valores recorridos ó representante (COLLAPSING_FIND) para evitar recorrer novamente a clase en caso dunha chamada posterior a FIND.

Como exemplos de elo, sexa o grafo

$$x_1 \to x_2 \to \ldots \to x_k \to u$$
$$y_1 \to y_2 \to \ldots \to y_j \to v, \quad k > j.$$

UNION(u, v) devolve
$x_1 \to x_2 \to \ldots \to x_k \to u \to y_1 \to y_2 \to \ldots \to y_j \to v,$

Figura 3.1: Unificación de termos en forma de grafos



mentras que UNION_WITH_WEIGHT(u, v) devolve
$y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_j \rightarrow v \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_k \rightarrow u$.

Neste mesmo exemplo, tense que FIND$(x_1) = u$ sen modificar o grafo e en cambio COLLAPSING_FIND(x_1) ó mesmo tempo que busca o representante u da clase, convirte o grafo

$$x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_k \rightarrow u \qquad \text{en}$$
$$x_1 \rightarrow u,\ x_2 \rightarrow u, \ldots, x_k \rightarrow u,$$

co que se acelera toda búsqueda posterior. Estas funcións UNION_WITH_WEIGHT e COLLAPSING_FIND son as respectivas UNION e FIND do algoritmo de Huet.

Bidoit e Corbin[17] propuxeron modificar o algoritmo inicial de Robinson, utilizando extensamente a representación por grafos. Xeralmente, neste tipo de representación dos termos, as variables ocupan lugares prefixados no grafo e todas as súas diferentes posicións corresponden a distintas líneas dos pais ás variables. E.g. $f(x, x)$ é representado por $\begin{array}{c} f \\ \downarrow\ \downarrow \\ x \end{array}$. Bidoit e Corbin empregan un tipo de grafos para os que esto mesmo se realiza para os subtermos non variables. Polo tanto o termo $f(gx, gx)$ correspóndese co grafo $\begin{array}{c} f \\ \downarrow\ \downarrow \\ g \\ \downarrow\ \downarrow \\ x \end{array}$ Desta forma, e aínda sen necesidade das melloras de Huet, conséguese reducir a complexidade do algoritmo de Robinson a un custe cadrático.

Algoritmo 3.4 ([17]) *Algoritmo de Bidoit e Corbin :*
Entrada : O par de termos s e t a unificar, expresado como grafo.
Saída : Certo se son unificables e o grafo resultante. O unificador máis xeral obtense en igual forma que no algoritmo anterior.

```
Unificar(s,t) =
  Se s = t Enton Certo
  Noutro caso
    Se t e no funcional Enton u = s, v= t
    Noutro caso u =t, v = s
      Se u e unha variable Enton
        Se APARECE(u,v) Enton Falso     /* APARECE realiza o chequeo de ciclos */
        Noutro caso
          Facer { UNION(u,v) e devolver Certo }
      Noutro caso                       /* u e v son nos funcionais */
        Se simbolo(u) ≠ simbolo(v) Enton Falso /* clash */
        Noutro caso UNION(u,v)
```

Sexa n = *aridade(simbolo(u))*
Sexan $(u_1, \ldots, u_n)$ = *hijos*(u)
Sexan $(v_1, \ldots, v_n)$ = *hijos*(v)
i = *0, bool = Certo*
Mentras $i < n$ *e bool*
Facer { i = *i+1, bool = unificar (FIND* (u_i), *FIND* (v_i)) }

Toda chamada a *unificar* ou ben devolve inmediatamente *Certo*, ou ben realiza un número de chamadas recursivas a *unificar* e, xa que o nó superior non é alcanzable só existe un número lineal de chamadas de FIND, UNION e APARECE.

A complexidade de FIND e de APARECE é lineal, polo que o custe total é cadrático. APARECE é claramente lineal.

No seguinte exemplo, pode observarse que o de FIND tamén pode chegar a ser lineal na lonxitude da entrada : $s = f(x_1, x_2, \ldots, x_{n-1}, y)$, $t = f(x_2, x_3, \ldots, x_n, x)$. Na execución de *unificar* (s, t) realízanse n chamadas a FIND e a i-ésima chamada ten que recorrer o grafo

$$x_{i-1} \rightarrow x_{i-2} \rightarrow \ldots \rightarrow x_1.$$

Ružička e Privara [158] modificaron o algoritmo de Corbin e Bidoit eliminando o test de ciclos (APARECE), por ser a causa do custe cadrático. En vez de elo, empregan, como no algoritmo de Paterson e Wegman, un marcador que indica que o nó xa foi procesado. Debido a que o marcador non permite detectar tódolos posibles casos de ciclos, colocan un novo test de ausencia destes ó final de todo o proceso como no algoritmo de Huet. Desta forma consiguen que o custe sexa semellante ó de Huet.

O algoritmo de Paterson-Wegman[137], é teóricamente o mellor por ter un custe lineal, se ben precisa un complexo sistema de punteiros entre os nós, o que o fai pouco práctico, dado que a lonxitude dos termos a unificar é habitualmente pequena. A entrada é un DAG *(directed acyclic graph)* con punteiros de pais a fillos e tamén dos fillos ós correspondentes pais.

Utilizan, ó igual que Huet [85], as relacións de equivalencia *válidas* que son aquelas para as cales :

i) non existen nós relacionados cuios símbolos sexan funcionais e distintos,

ii) os correspondentes fillos de dous nós equivalentes, tamén o son i.e. indicando a relación válida por $\approx$, se $u = f(u_1, \ldots, u_n) \approx v = f(v_1, \ldots, v_n)$ entón $u_i \approx v_i, 1 \leq i \leq n$.

As clases de equivalencia pódense ordenar a partir do orden parcial (pais-fillos).

O algoritmo de Paterson e Wegman integra a transmisión da relación válida co propio test de ciclos e devolve a sustitución nunha forma chamada *forma factorizada* (coñecida igualmente por *forma ordenada* ou tamén como *forma dag* ou

forma triangular). Esto significa que o unificador resultante obtense finalmente da composición de sustitucións $\{x_n/t_n\} \circ \ldots \circ \{x_1/t_1\}$ [3].

Os principais elementos do algoritmo de Paterson e Wegman son :

- a existencia de líneas entre nós que indican que están na mesma clase de equivalencia e que se transmiten cara abaixo relacionando os correspondentes fillos,
- o emprego dunha función *Finish* que é aplicada ós nós (una para cada nó da clase) e que realiza :
  - a conexión de nós equivalentes (menos o par inicial),
  - a transmisión da relación ós correspondentes fillos aplicando a propiedade *ii*),
  - o borrado de líneas e nós conectados co nó actual cando xa non son necesarios,
  - a aplicación de *Finish* nos predecesores dun nó. Cando *Finish* é aplicada a un nó, se este ten algún pai non borrado, aplícase inmediatamente *Finish* a ese pai antes de continuar co nó previo, para que poidan ser eliminados os seus predecesores. Este paso é clave na obtención dunha forma factorizada para a sustitución resultante e, no seu caso, na detección de ciclos.
- utiliza cinco punteiros para cada nó : símbolo do nó, fillos de este, pais do nó, sinalador de estado, líneas non dirixidas entre o nó e outros nós equivalentes.

De Champeaux, correxíu un erro do algoritmo de Paterson e Wegman e propuso algunhas modificacións como :

- utilizar líneas dirixidas entre nós equivalentes, o que é igualmente eficiente e en parte menos costoso que as non dirixidas,
- empregar un marcador de nós, en vez de eliminar materialmente os nós procesados,
- xerar unha sustitución resultante en vez dunha simple parada do algoritmo,
- procesar inicialmente os termos de entrada para obter os punteiros utilizados por Paterson e Wegman.

Algoritmo 3.5 ([137, 49]) *Algoritmo de Paterson e Wegman :*
Entrada : O par de termos s e t a unificar, expresado como DAG(Directed Acyclic Graph)
Saída : A sustitución Construír_Sigma

[3] neste mesmo capítulo aclárase o concepto de sustitución factorizada.

```
Unificar (s,t) =
  Crear unha linea entre s e t
  Mentras exista unha linea horizontal entre os nos u e v
    un de eles (e.g.u) non funcional
      Facer { Finish(u) }
  Mentras exista unha linea horizontal entre os nos u e v
    Facer { Finish(u) }
  Construir_Sigma(Variables)
```

Onde a funcion Finish se define como :

```
Finish(r) =
  Se Completo(r) Enton Fin
  Noutro caso
    Se punteiro(r) ≠ Nil Enton Fallo /* Non unificables */
    Noutro caso
      Sexa pila =∅
      Sexa punteiro(r) = r
      Push(r,pila)
      Mentras Pila non vacia
        Facer { Sexa s = Pop(Pila)
        Se s,r nos funcionais e operador-superior(s) ≠
          operador-superior(r) Enton Fallo /* Clash */
        Noutro caso
          Facer {
            Para cada no t, onde t = pai(s)
              { Facer Finish(t) }
            Para cada linea horizontal (s,u)
              Facer {
                Se completo(u) o ben u = r Enton Ignorar t
                Noutro caso
                  Se punteiro(u) = NIL Enton punteiro(u) = r
                    Push(u)
                  Noutro caso
                    Se punteiro(u) ≠ r Enton Fallo /* non unificables */
                    Noutro caso Ignorar t
                Borrar linea horizontal (s,u) }
            Se s≠ r Enton
              Se s·e variable Enton Subst(s) = r
                Engadir s a SIGMA
              Noutro caso Crear lineas { j-esimo-fillo(r), j-esimo-fillo(s)}
            Sexa completo(s) = certo }
```

}
Sexa completo(r) = certo

Escalada e Ghallab deron un novo algoritmo case lineal que está baseado no algoritmo case-lineal de Huet, buscando especialmente reducir o custe de inicialización e obter unha baixa complexidade en custe promedio, así como empregar un pequeno número de punteiros. Eliminan as regras de peso na operación de UNION e a cambio empregan unha serie de punteiros que permiten minimizar o número de operacións de búsqueda (tipo COLLAPSING-FIND) e móstrase explícitamente a sustitución resultante.

Como no de Paterson e Wegman, os ciclos detéctanse mediante un punteiro que se crea para tódalas variables que pertecen a unha clase procesada. Este mesmo punteiro é usado posteriormente para coñecer o valor da instanciación resultante para cada variable.

Algoritmo 3.6 ([60]) *Algoritmo de Escalada e Ghallab*
Entrada : O par de termos (s,t)
Saída : O unificador máis xeral $\{x/VERE(x))\}$

```
Unificar(s,t) =
  HERE(s,t)              / Homogeneus Equivalence RElation */
  para toda variable x de (s,t)
    Facer { VERE(x) }   /* Valid Equivalence Relation */
```

onde

```
HERE(s,t) =
  Se s e t non son variables identicas ou simbolos constantes iguais Enton
    Se s e variable Enton VAR-HERE(s,t)
    Noutro caso
      Se t e variable Enton VAR-HERE(t,s)
      Noutro caso
        Se operador superior(s) ≠ operador superior(t) Enton Fallo /* clash */
        Noutro caso para i = 1 ata aridade (operador superior (s))
          { Facer HERE(s_i, t_i) }
  Noutro caso Ignorar
```

Onde *HERE(s,t)* permite transmitir a equivalencia dos nós ós seus descendentes, e VAR-HERE fai modificar unha serie de punteiros chamando á función RECUR-HERE e ós procedementos *source* e *collapse*. Cando acaba de transmitirse HERE, a instanciación das variables é recollida na función *VERE* nunha forma algo semellante á empregada por Champeaux coa función *Ready*.

3.1.2 Algoritmos que utilizan a transformación de ecuacións.

O primeiro algoritmo de unificación (Herbrand) acepta como entrada un conxunto de ecuacións e realiza sucesivas transformacións ata obter unha forma sinxela. Martelli e Montanari [119] deseñaron un algoritmo que transforma multiecuacións e que é xeralmente recoñecido como heredero do de Herbrand[168].

A transformación de ecuacións, cambia o enfoque do problema de unificación. Aquí non é un valor booleano (ser unificables) e unha sustitución (solución) o que se extrae a partir dun conxunto de pares, senon un obxecto do mesmo tipo que a entrada, xa que se parte dun multiconxunto de ecuacións e obtense outro multiconxunto ó final do proceso. Esto permite integrar a unificación de pares de termos e a de conxuntos de pares. As transformacións utilizadas atópanse separadas da xestión das mesmas, polo que se pode estender a outros tipos de unificación (ecuacional [103], Xénero Ordenado [95], Orden Superior[73]) adaptando as regras e/ou a xestión ós diferentes casos.

Con esta perspectiva, a unificación sintáctica pódese ver como unha relación sobre multiecuacións coa propiedade de que todo problema é reducible a unha forma normal (única agás equivalencias) e a relación é finitaria.

As ecuacións irreducibles son : ben as ecuacións en *forma resolta*, ou na forma *factorizada resolta* dependendo dos algoritmos empregados.

As principais propiedades que relacionan as transformacións de ecuacións e as solucións son :

- *solidez*, se cada regra utilizada produce un novo conxunto cuias solucións están incluidas nas do de partida,
- *completitude*, cando as ecuacións irreducibles corresponden ás solucións do problema inicial,
- *terminación*, se o proceso detense logo dun número finito de pasos,
- *imparcialidade*, tódalas solucións do sistema inicial son realmente alcanzadas a partir de algunha transformación.

As transformacións usadas na unificación sintáctica verifican as catro propiedades así como a confluencia do conxunto de regras. Gracias a elo, a árbore de nós obtidos por diferentes transformacións a partir do nó inicial, non precisa ser estendido e pode escollerse un camiño arbitrario dende o problema inicial a un nó irreducible.

En moitos procesos de unificación faise preciso utilizar novas variables (non presentes no problema inicial) para as cales non é importante coñecer as súas instanciacións senon só a existencia dalgunha instanciación que resolve o problema dado. Por elo, soen ser tratadas en diferente forma que o resto de variables [44].

Definición 3.1 *Unha* multiecuación *é un multiconxunto de pares* $\{(s_1,t_1),\ldots,(s_n,t_n)\}$ *onde par* (s_i,t_i) *pódese atopar repetido*[4]. *Tamén se*

[4]en [119] unha multiecuación é un conxunto de pares (Variables, Termos), e utilízase unha variable extra para representar os termos iniciais que deben ser unificados.

chamará multiecuación á expresión : $\Gamma = \exists z_1, z_2, \ldots, z_m \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$. *As variables* $z_1, \ldots, z_n$ *dirase que se atopan* acotadas. *As variables de* Γ *non acotadas son as* variables libres *de* Γ.

Definición 3.2 *Sexa* S *un conxunto o un multiconxunto de termos e* θ *unha sustitución. Defínese a aplicación de* θ *en* S *por :* $S\theta = \{s\theta, s \in S\}$.

Se S *é un conxunto ou multiconxunto de pares, dise que* θ *unifica* S *se* θ *unifica tódolos pares de* S.

No caso de que aparezan variables acotadas, considérase como unificador de $\exists z_1, \ldots, z_n, s \stackrel{?}{=} t$ *toda sustitución* σ *tal que se* $Z = \{z_1, \ldots, z_n\}$, *existe* σ', $\sigma_{V \setminus Z} = \sigma'_{V \setminus Z}$, *e* $s\sigma' = t\sigma'$.

Definición 3.3 *Sexan* $\Gamma = \exists z_1, z_2, \ldots, z_m \{s_1 \stackrel{?}{=} t_1, \ldots s_n \stackrel{?}{=} t_n\}$, $\Delta = \exists z'_1, z'_2, \ldots, z'_{m'}$, $\{s'_1 \stackrel{?}{=} t'_1, \ldots s'_{n'} \stackrel{?}{=} t'_{n'}\}$. *Ambas multiecuacións* Γ *e* Δ *son* equivalentes *se :*

- *para toda sustitución* θ *tal que* $Dom(\theta) \cap \{z_1, \ldots, z_m, z'_1, \ldots, z'_{m'}\} = \emptyset$, *verifícase que :* θ *unifica* Γ *se e só se unifica a* Δ,
- *as variables libres de* Γ *e de* Δ *son as mesmas.*

Solucións en forma resolta

Definición 3.4 *A multiecuación* $\Gamma = \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$, *está en* forma resolta *se :*

- $\forall i, j,\ 1 \leq i < j \leq n,\ x_i \neq x_j$,
- $\forall i, j,\ 1 \leq i, j \leq n, i \neq j,\ x_i \notin Var(t_j)$,
- *se* $x_i \in Var(t_i)$ *e* $t_i = x_i$, *entón* $\forall j \neq i,\ x_i \notin Var(t_j)$.

Se $\Gamma = \exists z_1, z_2, \ldots, z_m \{x_1 \stackrel{?}{=} t_1, \ldots x_n \stackrel{?}{=} t_n\}$ *dise que está en forma resolta se amais das condicións anteriores verifícase que :*

- $\forall i, j\ \ 1 \leq i \leq n, 1 \leq j \leq m,\ x_i \neq z_j$,
- $\forall i, j\ \ 1 \leq j \leq m, \exists i\ 1 \leq i \leq n,\ z_j \in Var(t_i)$.

Dada unha multiecuación en forma resolta $\Gamma = \exists z_1, z_2, \ldots, z_m \{x_1 \stackrel{?}{=} t_1, \ldots x_n \stackrel{?}{=} t_n\}$ obtense un unificador máis xeral idempotente $\sigma_\Gamma = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$.

Na unificación sintáctica, non é necesario utilizar variables acotadas xa que tódalas variables son tratadas de igual xeito por pertecer ó problema inicial.

O seguinte conxunto de transformacións, que produce unha forma resolta equivalente a unha multiecuación dada, é coñecido como algoritmo de Herbrand-Robinson [2] e aparece a veces sinalado na unificación ecuacional como $\mathcal{ST}$ (*syntactic transformations*):

Borrado $\Gamma \cup \{s \stackrel{?}{=} s\} \Rightarrow \Gamma$.

Descomposición $\Gamma \cup \{f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)\} \Rightarrow \Gamma \cup \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$
para todo $f \in \Sigma$.

Detección de conflictos $\Gamma \cup \{f(s_1, \ldots, s_n) \stackrel{?}{=} g(t_1, \ldots, t_m)\} \Rightarrow Fallo$
se $f \neq g$ ou $f = g$ e $n \neq m$ con $f, g \in \Sigma$.

Pegado de clases $\Gamma \cup \{x \stackrel{?}{=} y\} \Rightarrow (\Gamma\{x/y\}) \cup \{x \stackrel{?}{=} y\}$
se $x, y \in V$, $x \neq y$.

Chequeo de ciclos 1 $\Gamma \cup \{x \stackrel{?}{=} s\} \Rightarrow Fallo$
se $x \in Var(s)$ e $x \neq s$.

Eliminación de variables $\Gamma \cup \{x \stackrel{?}{=} t\} \Rightarrow (\Gamma\{x/t\}) \cup \{x \stackrel{?}{=} t\}$
se $x \notin Var(t), x \in Var(\Gamma)$.

As regras de *Pegado de clases* e *Detección de conflictos*, aínda que parecen idénticas, mostran a diferencia entre instanciacións variables e non variables. As regras *Chequeo de ciclos 1* e *Detección de conflictos* devolven *Fallo* como a forma irreducible dun sistema sen unificadores e.g. un par de constantes diferentes $a \stackrel{?}{=} b$.

O algoritmo determinístico de Robinson pode considerarse como unha derivación do anterior cando as ecuacións son almacenadas nunha pila e as transformacións son aplicadas sucesivamente ós distintos elementos da pila.

A unificación é unha operación conmutativa. Se se consideran idénticos os pares $s \stackrel{?}{=} t$ e $t \stackrel{?}{=} s$, a aplicación da regra *Pegado de clases* provoca a non terminación. Unha solución é usar un orden total $>$ nas variables, de forma que sempre sexa reemplazada a variable maior pola menor. Outra posibilidade é a de considerar os pares ordenados (e.g.[45]) e aplicar a transformación $\{s \stackrel{?}{=} x\} \Rightarrow \{x \stackrel{?}{=} s\}$ se $s \notin V$ ou ben desdoblar as regras *Chequeo de ciclos 1* e *Eliminación de variables.*

O Teorema 1 de [95] non é totalmente coherente coa definición de forma resolta para Γ, xa que a eliminación daquelas variables de Γ que só aparecen en expresións eliminables pola regra *Borrado* impide que as variables de Γ e da súa forma resolta sexan as mesmas.

Exemplo 3.3 *Aplicar $\mathcal{ST}$ a $\Gamma = \{f(x, y) \stackrel{?}{=} f(x, z)\}$ produce $\Delta = \{y \stackrel{?}{=} z\}$ pero pola definición de forma resolta empregada en [95] e xa que Δ non ten as mesmas variables que Γ, non pode ser a sua forma resolta.*

Unha forma de evitar este problema é non empregar a operación de *Borrado* se fai desaparecer unha variable non existente en nengún termo do resto das ecuacións.

Solucións en forma factorizada

Jouannaud e Kirchner[95] chamaron *forma dag resolta* (*forma triangular* en [184]), ás ecuacións correspondentes a sustitucións factorizadas. Dado un problema de unificación, pódese obter unha árbore deste tipo sen necesidade de transmitir a instanciación (mover termos non variables).

Definición 3.5 *Sexa* $\Gamma = \exists z_1, z_2, \ldots, z_m \{x_1 \stackrel{?}{=} t_1, \ldots x_n \stackrel{?}{=} t_n\}$. Γ *está en* forma factorizada resolta *se :*

- $\forall i, j,\ 1 \leq i < j \leq n,\ x_i \neq x_j$,
- $\forall i, j,\ 1 \leq i < j \leq n,\ x_i \notin Var(t_j)$,
- $\forall i,\ 1 \leq i \leq n$, *se* $t_i \in V$, *entón* $\forall j,\ 1 \leq j \leq m,\ t_i \neq z_j \neq x_i$, *se* $t_i = x_i$ *entón* $\forall k \neq i,\ x_i \notin Var(t_k)$,
- $\forall j,\ 1 \leq j \leq m,\ \exists i, 1 \leq i \leq n,\ z_j \in Var(t_i)$.

Se Δ é unha forma factorizada equivalente a Γ non compre que Δ sexa unha forma factorizada obtida a partir da multiecuación Γ, no senso de que poido pasarse de Γ a Δ transmitindo a instanciación para algunha variable de Γ.

Exemplo 3.4 $\Gamma = \{x \stackrel{?}{=} f(fz),\ y \stackrel{?}{=} fz,\}$ *é unha forma factorizada resolta para o problema* $\Gamma' = \{g(x) \stackrel{?}{=} g(f(f(z)),\ g(y) \stackrel{?}{=} g(fz)\}$ *pero nono é para a ecuación equivalente* $\Delta = \{g(x) \stackrel{?}{=} g(f(y)),\ g(y) \stackrel{?}{=} g(f(z))\}$.

Precísase polo tanto aclarar o significado da forma factorizada para unha multiecuación dada.

Definición 3.6 ([119]) *Sexa* $\Gamma = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$ *unha multiecuación. Defínense as relacións seguintes entre as variables de* Γ *:*

$x \sim y$ *se* $\exists i, \pi\ s_i/\pi = x,\ t_i/\pi = y$,

$\simeq$ *clausura reflexiva, simétrica e transitiva de* $\sim$,

$x \prec y$ *se* $\exists i, \pi,\ s_i/\pi = x, y \in Var(t_i/\pi), y \neq t_i/\pi$ *ou* $t_i/\pi = x, y \in Var(s_i/\pi), y \neq s_i/\pi$,

$\prec_{\simeq}$ *é a relación* $\prec^+$ *estendida ás clases de equivalencia* $[]_{\simeq}$. *Esto é* $[x]_{\simeq} \prec_{\simeq} [y]_{\simeq}$ *sse* $\exists x' \in [x]_{\simeq}, y' \in [y]_{\simeq}, x' \prec^+ y'$.

Exemplo 3.5 *E.g. en* $\{f(x,x,x) \stackrel{?}{=} f(y, gz, u)\}$ *tense* $x \sim y$, $y \simeq u$, $x \prec z$, $[y]_{\simeq} \prec_{\simeq} [z]_{\simeq}$.

Definición 3.7 *Sexan* $\Gamma = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n = t_n\}$, $\Delta = \{x_1 \stackrel{?}{=} w_1, \ldots, x_m \stackrel{?}{=} w_m\}$ *onde* Δ *é unha forma factorizada resolta equivalente a* Γ. Δ *é unha* forma factorizada resolta respecto *de* Γ *se ambas son equivalentes e para toda multiecuación* $x_i \stackrel{?}{=} w_i \in \Delta, \exists j, \pi, 1 \leq j \leq n, \pi \in \Pi(s_j), \pi \in \Pi(t_j)$ *tal que* $s_j/\pi = y, y \simeq x$, $tam\_estr(w_i) \leq tam\_estr(t_j/\pi)$ [5] *(onde a relación* $\simeq$ *establécese a partir da multiecuación* Γ *e os pares están tomados sen orientación).*

Exemplo 3.6 *Sexa*
$\Gamma = \{f(x_1, f(x_2, f(\ldots f(x_{n-1}, x_n))\ldots) \stackrel{?}{=} f(f(x_2, x_2), f(\ldots f(f(x_{n-1}, x_n), x_n)\ldots)$. *Na figura 3.2 pode observarse : unha representación deste problema (A), en B) unha forma resolta e en C) móstrase unha forma factorizada resolta respecto de* Γ.

Modificando o conxunto de transformacións $\mathcal{ST}$, obtense un algoritmo que proporciona unha forma factorizada resolta :

Borrado $\Gamma \cup \{s \stackrel{?}{=} s\} \Rightarrow \Gamma$.

Descomposición $\Gamma \cup \{f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)\} \Rightarrow \Gamma \cup \{s_1 \stackrel{?}{=} t_1, \ldots s_n \stackrel{?}{=} t_n\}$
para todo $f \in \Sigma$.

Detección de conflictos $\Gamma \cup f(s_1, \ldots, s_n) \stackrel{?}{=} g(t_1, \ldots, t_m) \Rightarrow Fallo$
se $f \neq g$ ou $f = g$ e $n \neq m$ con $f, g \in \Sigma$.

Pegado de clases $\Gamma \cup \{x \stackrel{?}{=} y\} \Rightarrow (\Gamma\{x/y\}) \cup \{x \stackrel{?}{=} y\}$ se $x, y \in V, x \neq y$.

Chequeo de ciclos 2 $\Gamma \cup \{x_1 \stackrel{?}{=} s_1[x_2], x_2 \stackrel{?}{=} s_2[x_3], \ldots, x_n \stackrel{?}{=} s_n[x_n]\} \Rightarrow Fallo$
se $\exists i,\ 1 \leq i \leq n, s_i \neq x_{i+1}$.

Factorización $\Gamma \cup \{x \stackrel{?}{=} s, x \stackrel{?}{=} t\} \Rightarrow \Gamma \cup \{x \stackrel{?}{=} s, s \stackrel{?}{=} t\}$ se $0 < |s| \leq |t|$.

As observacións anteriores acerca da regra *Pegado de clases* e *Borrado*, seguen a ser aplicables a este caso. Son evitadas mediante a incorporación de :

Conmutar_termos $\Gamma \cup \{s \stackrel{?}{=} x\} \Rightarrow \Gamma \cup \{x \stackrel{?}{=} s\}$ se $x \in V, s \notin V$.

e sustituíndo *Borrado* por :

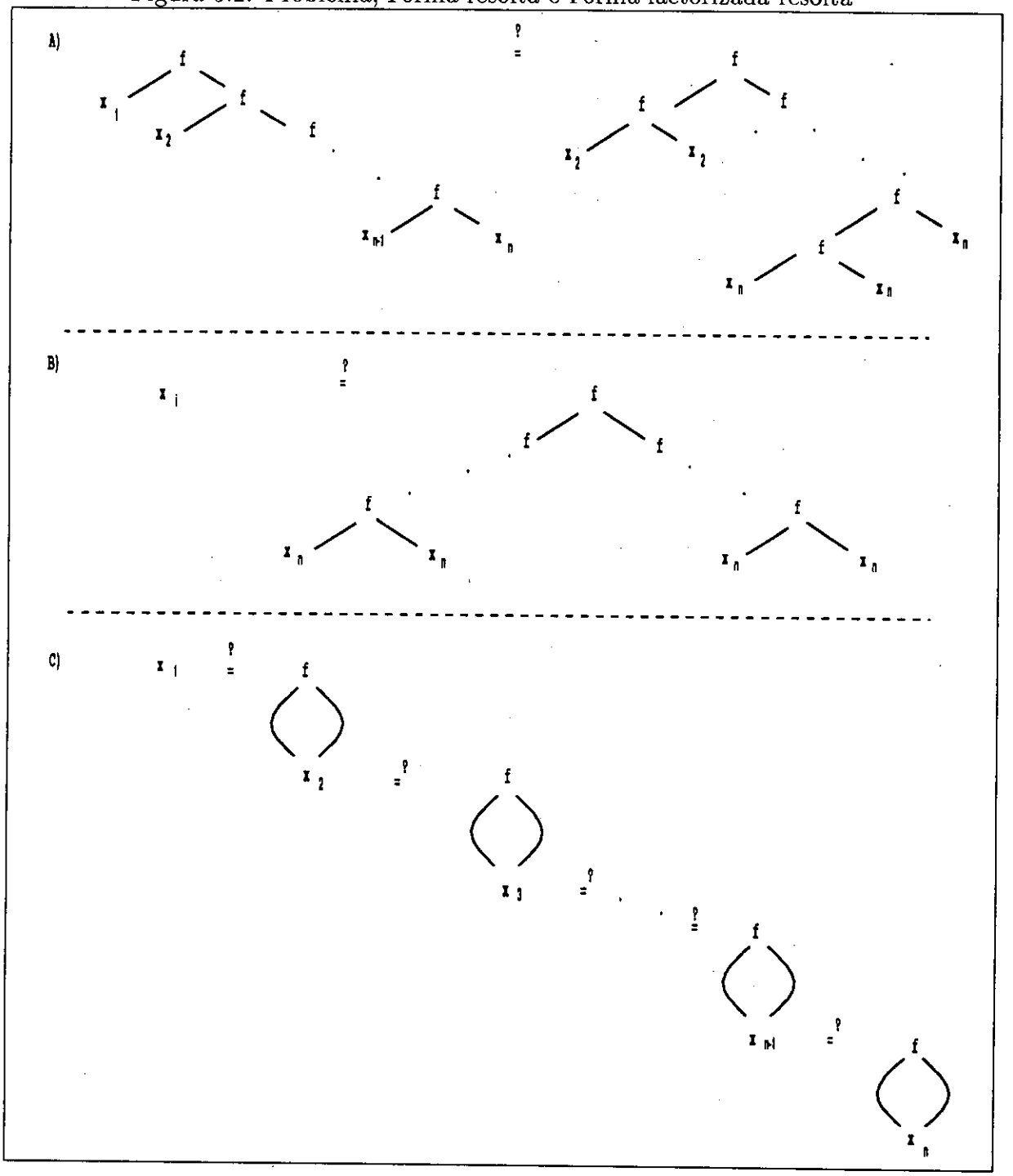
Borrado 2 $\Gamma \cup \{s \stackrel{?}{=} s\} \Rightarrow \Gamma$ se $s \in V$ e $s \in Var(\Gamma)$

Como se indicou previamente, *Conmutar_termos* pode ser sustituida pola duplicación de *Eliminación de variables* e *Factorización* atendendo á posición na que se atope a variable (en s ou en t).

O seguinte resultado é unha adaptación do teorema *3.3* de Martelli e Montanari [119].

[5] con esta condición asegúrase que non existe incremento dos operadores.

Figura 3.2: Problema, Forma resolta e Forma factorizada resolta



3.2 *Sexa* Γ *unha multiecuación con unificador, existe unha forma factorizada resolta respecto* Γ.

Proba. Se o conxunto de variables non é vacío, necesariamente deben de existir clases de variables $[x]_{\simeq}$ minimais, pois noutro caso existe un ciclo en Γ debido a que os unificadores de Γ igualan os subtermos da forma $x, t[y], t'[x], y$ con t ou t' non variables

.

Sexa x unha variable dunha clase minimal. Se $s_i/\pi = x, \pi \in \Pi(t_i)$, os operadores en $s_i/\pi', t_i/\pi'$ con $\pi' \prec \pi$ deben coincidir xa que Γ ten unificadores. Aplícanse as regras *Descomposición* e *Pegado de clases* a tódalas ecuacións e ás novas ecuacións derivadas e obtense un sistema da forma $\Gamma' = \{x_1 \stackrel{?}{=} x, \ldots, x_k \stackrel{?}{=} x\} \cup \{x \stackrel{?}{=} w_1, \ldots, x \stackrel{?}{=} w_j\} \cup \{s'_1 \stackrel{?}{=} t'_1, \ldots, s'_l \stackrel{?}{=} t'_l\}$, onde as variables $x, x_1, \ldots, x_k$ non están presentes en nengunha outra parte de Γ' pola minimalidade da clase $[x]$. Se $j > 1$ aplícase a regra *Factorización* en $\{x \stackrel{?}{=} w_1, \ldots, x \stackrel{?}{=} w_j\}$, ata obter un sistema $\Gamma'' = \{x_1 \stackrel{?}{=} x, \ldots, x_k \stackrel{?}{=} x, x \stackrel{?}{=} w_1\} \cup \{s''_1 \stackrel{?}{=} t''_1, \ldots, s''_{l+j-1} \stackrel{?}{=} t''_{l+j-1}\}$. O resultado obtense ahora aplicando inducción no número de variables da parte non resolta $\{s''_1 \stackrel{?}{=} t''_1, \ldots, s''_{l+j-1} \stackrel{?}{=} t''_{l+j-1}\}$ □.

A selección das ecuacións con variables maximais esixe dispor dunha forma de coñecer estas variables e por tanto das ecuacións a seleccionar, o que se coñece como *oráculo*. O algoritmo de $\mathcal{ST}$, cando se supón que ten un oráculo sen custe adicional dise que é un *algoritmo de Herbrand con oráculo*[2].

O algoritmo de Martelli e Montanari, é innovador tanto por empregar a transformación de multiecuacións, como por usar un oráculo de baixo custe para detectar as variables minimais e amais proporciona unha forma extremadamente factorizada do unificador. O oráculo vén dado por un contador de posicións de cada variable (concretamente de cada conxunto de variables na mesma clase) para o que se precisa recorrer unha sola vez tódolos termos antes de iniciar o proceso de unificación e sendo o seu custe posterior moi reducido.

Neste algoritmo, a forma factorizada obtense extraendo a *parte común* e a *fronteira* do conxunto de multiecuacións mediante a operación de *reducción*. A parte común dun conxunto de termos, expresa o termo máis xeral de ese conxunto, mentras que a fronteira reúne as ecuacións obtidas ó extraer a parte común. Na operación de compactación xúntanse todas as ecuacións que se encontran na mesma clase (por $\simeq$).

Definición 3.8 *Sexan* s, t *termos. Defínese a parte común Com(s,t) e a fronteira Front(s,t) por :*

- *se s e t son variables,* $Com(s,t) = s$, $Front(s,t) = ([s,t], \emptyset)$,
- *se s é variable e t nono é,* $Com(s,t) = s$, $Front(s,t) = \{([s], t)\}$ *(análogamente se* $t \in V$*),*
- *se* $s = f(s_1, \ldots, s_n), t = f(t_1, \ldots, t_n)$, $Com(s,t) = f\big(Com(s_1,t_1), \ldots, Com(s_n,t_n)\big)$, $Front(s,t) = \bigcup_{1 \le i \le n} Front(s_i, t_i)$,

- *noutro caso Com(s,t) e Front(s,t) non están definidos.*

Algoritmo 3.7 *Unificación de Martelli e Montanari :*
Entrada : O par $(U, \emptyset)$ con U = conxunto inicial de multiecuacións.
Saída : $(\emptyset, T)$ onde T é o conxunto de multiecuacións en forma factorizada.

(1) Mentras U sexa non vacio,
(1.1) Seleccionar unha multiecuacion (S, M) de U, tal que os elementos de S non aparecen en nengunha outra multecuacion.
Se tal multiecuacion non existe Fallo / Ciclo */*
(1.2) Se M e vacio
Enton pasar a multiecuacion de U o final de T.
Noutro caso
Facer {
(1.2.1) Computar a parte comun e a fronteira de M.
Se M non ten parte comun Fallo / Clash */*
(1.2.2) Transformar U usando a reduccion e a compactacion
(1.2.3) Pasar a multiecuacion $S = C$ dende U o final de T } .

(2) Parar con exito.

Onde a compactación é a regra que se ten denominado *Pegado de clases*, e a reducción é **Reducción** $\Gamma \cup \{x \stackrel{?}{=} s_1, x \stackrel{?}{=} s_2, \ldots, x \stackrel{?}{=} s_n\} \Rightarrow \Gamma \cup \{x \stackrel{?}{=} c\} \cup \{c \stackrel{?}{=} s_1, \ldots, c \stackrel{?}{=} s_n\}$ se $2 \leq n, 1 \leq i \leq n, x \notin s_i, s_i \notin V$, sendo c a parte común dos $s_i, 1 \leq i \leq n$ operación similar á *Factorización.*

Algunhas das características destacadas do algoritmo de Martelli-Montanari son:

- integración da factorización da sustitución co test de ciclos, mediante o contador de presencia das variables nos termos (M) das multiecuacións. Esto permite coñecer o orden en que deben ser eliminadas as variables. Tén certa similitude co algoritmo de Paterson e Wegman,

- a reunión de clases con operadores (compactificación). En canto se reune unha clase, procédese ó chequeo completo dos operadores,

- cando dous conxuntos de variables se xuntan, o resultado é unha nova lista obtida pola adición da menor á maior, para o que se emprega un contador. Sen este contador a complexidade chegaría a ser cadrática no peor caso. Tén algún paralelismo coa operación UNION-WITH-WEIGHT empregada no algoritmo de Huet,

- o custe teórico do algoritmo é $\mathcal{O}(n \cdot log(n))$ (algo superior a Huet e Paterson e Wegman), se ben a preparación inicial é máis sinxela que nos outros dous.

Dada a unicidade das formas resoltas, a relación $\Rightarrow$ é finitaria e confluente polo que o orden empregado para aplicar as regras non é importante para o resultado final. Pero poden existir diferencias importantes en canto a custe en tempo se se emprega unha ou outra estratexia, como fixeron Martelli e Montanari para a detección rápida de ciclos.

Colocar as regras de detección de fallo, o antes posible, evita un procesamento innecesario das ecuacións. Por elo, no control usual aplícanse reiteradamente a *Descomposición* e *Detección de conflictos* previamente ó resto de operacións.

Usando o orden :

$\Big(\big(\big(\,$ *Descomposición* $\cup$ *Detección_conflictos* $\,\big)^* \cup$

$(\,$ *Factorización* $\cup$ *Pegado de clases* $\,)^*\big)^* \cup (\,$ *Borrado* $\cup$ *Chequeo_ciclos* $\,)^*\Big)^*$ obtense un algoritmo que, como o de Huet[85], é moi eficaz na detección dos conflictos de operadores.

O unificador resultante dunha forma factorizada $\Delta = \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$ obtense por composición $\sigma_n \ldots \sigma_1$ onde $\sigma_i = \{x_i/t_i\}$. Esto mostra a forte compoñente secuencial da unificación dado que a última instanciación obtida $\{x_n/t_n\}$ debe ser a primeira en ser utilizada. A partir da sustitución factorizada $\delta = \{x_1/t_1, \ldots, x_n/t_n\}$, a sustitución σ obtense como o punto fixo de δ, esto es, $\delta^{m+1} = \delta^m = \sigma$, que é xustamente a sustitución idempotente obtida a partir de δ.

3.1.3 Comparación entre os algoritmos

Para realizar unha comparación entre os diferentes algoritmos, establécense os seguintes criterios de eficacia :

1. menores custes de espacio e tempo en función do tamaño da entrada,
2. simplicidade e baixo custe de preparación,
3. rendemento en probas con exemplos representativos do seu uso na práctica,
4. menor custe medio na unificación de termos construidos cun certo número de operadores e variables,
5. maior utilidade práctica (constatada polo uso).

1. Xa se indicou que o mellor é o de Paterson e Wegman. Existen outros algoritmos tamén de custe case lineal ou lineal : Robinson, Huet, Martelli e Montanari, D. Kapur,M.S. Krishnamoorthy e P. Narendran, Martelli, Escalada e Ghallab, Ružička e Privara. Entre os tres primeiros, se a probabilidade de conflicto é alta, o mellor algoritmo é o de Huet seguido do de Martelli e Montanari e se a probabilidade de ciclo é superior á de conflicto o éxito é superior o de Paterson e Wegman seguido do de Martelli e Montanari.

2. O mellor neste senso, é indudablemente o algoritmo de Herbrand-Robinson, que non precisa encabezamento xa que non dispón de estructuras adicionais.

   Dos tres algoritmos comparados, o de menor custe de preparación é o de Martelli e Montanari, pois a estructura de multitermos é bastante semellante á de pares de termos e, á vez que se recorren éstos para inicializar o contador de variables, realízase o chequeo de *clash*. O de maior custe de preparación é o de Paterson e Wegman. Debe notarse que a mellora proposta por D. de Champeaux, utiliza aínda un maior número de punteiros.

3. No artículo de Martelli e Montanari[119], cítase un traballo de Trum e Winterstein onde, implementados os tres en Pascal e con cinco tipos de datos a unificar, o de menor tempo promedio foi o de Martelli e Montanari, aínda que difiren bastante nos distintos casos.

   Escalada e Ghallab [60] compararon o seu algoritmo co de Martelli e Montanari mediante unha serie de problemas recollidos fundamentalmente de aplicacións típicas de proba automática e usando representacións bastante parecidas na lingoaxe LISP. Os experimentos realizados deron unha relación de tempos favorable ós primeiros inferior a 0,5 e incluso inferior a 0,3 para a maioría dos casos examinados.

   Ružička e Privara [158] compararon o seu algoritmo co de Corbin e Bidoit para varios tipos de problemas, resultando mellor o algoritmo de Ružička e Privara na maior parte dos casos. Xa que unha parte do chequeo de ciclos non se realiza ata o final, resulta máis lento nos casos nos que este test falla.

4. Respecto do custe promedio dos diversos algoritmos, L. Albert *et. al.*[2] estudiaron mediante análisis asintótico, o custe medio dos algoritmos de unificación en determinados casos, chegando ás seguintes conclusións:

   - o custe exponencial do algoritmo de Herbrand-Robinson dáse en situacións excepcionais : A complexidade media no caso dun só símbolo funcional de aridade 2 e dúas variables é lineal,
   - en situacións máis xerais (operadores binarios, constantes e variables) e se o algoritmo de Herbrand-Robinson modifícase demorando a eliminación de variables mentras poida aplicarse a descomposición ou detección de conflicto, o custe medio do algoritmo sobre o conxunto de pares é constante. Elo é así pola alta proporción de conflictos e a cada vez menor (en función da lonxitude dos termos) influencia dos pares unificables[6],
   - cando se dispón do algoritmo de Herbrand-Robinson e supoñendo un oráculo sen custe extra, o custe medio é novamente constante, e o custe promedio descende lixeramente do anterior, xa que non precisa o chequeo de ciclos,

[6] no citado traballo se mostra que o número destes pares é exponencialmente despreciable.

- se se utiliza o algoritmo de Martelli e Montanari o custe medio é lineal na lonxitude da entrada, aínda demorando o conteo das posicións de cada variable, ou sexa que o conteo se realice logo da primeira descomposición e chequeo de conflictos.

5. O algoritmo de Robinson pasou de ser o algoritmo de unificación por excelencia, a ser visto como un algoritmo cun custe desproporcionado, crítica hoxe considerada como pouco xusta, xa que o custe dun algoritmo non depende só do custe no seu peor caso. De feito, o seu uso é común en importantes procesos de unificación[144]. Na práctica, a lonxitude usual dos termos que son unificados non é excesiva (aínda que a signatura teña un gran número de operadores) o que relativiza máis o posible custe exponencial. Aínda así, a unificación é unha operación costosa na que algunhas melloras teóricas póden non ser prácticas polo seu custe de preparación.

   Na maioría das extensións da unificación precísanse algoritmos clásicos de unificación de termos libres. Meseguer *et al.* e Williams [124, 184] utilizan variantes directas do algoritmo de Martelli Montanari. Jaffar [92] deu un algoritmo de unificación para termos racionais (admite ciclos) que tamén é herdeiro directo do de Martelli e Montanari, xa que extrae a parte común e a fronteira, ten unha forma semellante de separar variables e termos, emprega un contador do número destas e segue ese esquema, se ben non precisa elexir a clase de variables maximais. Jaffar compárao en diversos problemas con outros algoritmos de unificación racionais e obtén úns resultados superiores ós algoritmos de unificación racional de Mukai e de Colmerauer [130, 42]. Jaffar considera un mérito do seu algoritmo, compartido co de Martelli e Montanari, que poida realizarse directamente a unificación de máis de dous termos conxuntamente. Non sinala, que esta unificación múltiple atópase xa no algoritmo de unificación inicial de Robinson.

3.2 Formas de acelerar a unificación

Como mostra da importancia do proceso de unificación, tense estimado que o custe da unificación na programación habitual en PROLOG supera o 50% do tempo de execución ([166]). Por elo, resulta fundamental atopar outras formas de acelerala. Entre estas se encontran :

- a implementación directamente en hardware,
- a paralelización da unificación,
- as técnicas de indexación dos símbolos e estructura dos termos.

3.2.1 Máquinas de Unificación e Técnicas de Codificación

O co-procesador máis coñecido para a compilación directamente da unificación en silicio é SUMA [154]. Outras unidades realizadas o diseñadas son UNIF de S. Chang para executar o algoritmo de Robinson, HUU (Hardware Unification Unit) desenrolado por Woo e UNIFIC de Gollakota [39, 78].

A máquina de WARREN [182] (WAM), é a máis coñecida das que, mediante un conxunto de instruccións en código máquina, compilan unha lingoaxe de programación lóxica (PROLOG). Existen outros códigos onde a unificación atópase incorporada por microinstruccións como KLO e PLM. Xa que estes conxuntos de instruccións son directamente realizables por hardware axeitado, pódense ver como máquinas de unificación.

Unha das técnicas de acelerar o proceso de unificación é a de indexación dos símbolos e estructura para comprimir o espacio empregado para gardar as cláusulas e acelerar a búsqueda e recuperación dos datos. A maioría das implementacións de PROLOG, indexan algúns datos como o predicado da cabeza dunha cláusula, o número de argumentos e algunha información acerca do tipo de datos (e.g. se son constantes, variables, funcións, listas, estructuras). Deste xeito realízase máis rápidamente un primeiro chequeo de conflictos.

Pese a elo, cando o número de cláusulas é moi elevado (poidendo chegar a ser de miles e decenas de miles, como nalgúns sistemas expertos [122]) é preciso reducir o tamaño ocupado por cada termo e simplificar a forma de comprobar a igualdade entre símbolos, aída que como consecuencia, se produzcan algúns *false drops* (éxitos que nono debían ser). A compresión pode realizarse codificando os argumentos, ou ben por unha codificación máis forte da estructura como as técnicas de codificación superimposta (na que os diferentes argumentos se encontran reunidos nun mesmo campo).

Na codificación dos argumentos [185], os datos gárdanse empregando todos eles un número fixo de símbolos 1. Se n é o número de bits reservado, desta forma pódense gardar $\binom{n}{n/2}$ datos. Unha posterior compresión permite reducir o número de bits a $log_2 \binom{n}{n/2}$. A ventaxa de empregar un número fixo de ceros e úns é que a búsqueda pode facerse máis efectiva, xa que se detecta un conflicto se *algún* 1 dos argumentos do termo inicial, correspóndese cun 0 dese argumento do termo almacenado. O inconveniente é que se poden producir falsos conflictos por utilizar termos cunha estructura máis complexa da prevista. Precisa unha plantilla para os datos (cláusulas e termos) e debe seleccionarse un número de bits dado para cada argumento, dependendo este do número de símbolos empregados e da profundidade esperada para cada subtermo.

Exemplo 3.7 *Para un termo da forma $f(a, g(b, c), h(d))$ son necesarios 36 bits se se reservan 6 bits para cada campo. Esto permite empregar ata*

$\binom{6}{3}$ *símbolos diferentes. O termo* $f(a, g(b, x), y)$ *pódese gardar como* 111000 010101 101010 111111 111111 111111, *mostrando que as variables non teñen nengún custe adicional.*

Na codificación superimposta, tódolos argumentos (como contido e posición) son gardados conxuntamente nun único campo cun pequeno número de símbolos 1 para cada argumento-posición, obtendo o código final do termo mediante a reunión (OR) de tódolos códigos dos argumentos. Poden producirse falsos éxitos debidos a que en ocasións, distintos argumentos teñen que compartir algúns bits. Na *codificación superimposta con representación externa para as variables*, emprégase unha plantilla para fixar as posicións e se precisa un campo para gardar os datos e un campo adicional de 1 bit para cada posición. Este último campo toma o valor 1 se nesa posición se atopa unha variable. A compresión e flexibilidade é maior que coa codificación dos argumentos, pero a cambio poden producirse máis conflictos non detectados debido a que na superimposición poden interferir as codificacións dos distintos atributos. O proceso de chequeo encarécese xa que debe observarse se nas posicións hai unha variable correspondente a esa posición. Con esta codificación tense realizado o NU-PROLOG [145].

Na codificación superimposta con representación interna para as variables, codifícanse tamén as posicións. As variables gárdanse mediante o código da posición. Desta forma conséguese unha maior flexibilidade e só se precisa estimar a profundidade da estructura e o número de símbolos. O custe de chequear cada posición é máis alto que nos outros casos, pero ten unha maior capacidade de almacenamento e de flexibilidade[43].

3.2.2 Unificación Paralela

As esperanzas postas no uso de técnicas de paralelización con vistas a reducir fortemente o custe de a unificación sintáctica, foron radicalmente frustradas polo resultado de Dwork *et al.*[57]. Sen embargo Vitter e Simons [181] mostraron como podían obterse importantes melloras na unificación aplicando extensamente a paralelización. Por elo continuóuse a investigación de detección de problemas de unificación que son paralelizables e a desenrolar mecanismos de unificación paralelos, ben en forma independente do seu uso, como no caso de ser usado como un coprocesador, ou ben integradamente dentro dun sistema (e.g. de proba automática) paralelo.

Definición 3.9 *Sexa* Γ *unha clase de problemas de lonxitude* n. Γ *está na clase :*

- *PTIME, se se pode resolver mediante un algoritmo, determinístico con custe polinómico en* n.

- *NC se empregando un número de procesadores polinómico en n, pode resolverse en* $log(\mathcal{O}(n))$,
- NC^2 *é a subclase de NC que contén ós problemas que se poden resolver en* $log^2\mathcal{O}(n)$ *usando suficiente número (polinómico en n) de procesadores,*
- *PC se* $\lim_{n\to\infty} \frac{Tempo_{PRAM}}{Tempo_{RAM}}$ *existe e tende a cero cando o número de procesadores é polinómico en n. onde* $Tempo_{PRAM}$ *é o tempo empregado en resolver o problema usando un programa baseado nun modelo de computación PRAM (uniform parallel random access machine), mentras que* $Tempo_{RAM}$ *éo para un programa secuencial usando un procesador RAM,*
- *PC* se para cada n existe un número de procesadores* $P(n)$ *tal que* $\frac{Tempo_{PRAM} \times P(n)}{Tempo_{RAM}} = \mathcal{O}(n)$,
- *é* log-space complete *para a clase H, se todo problema de H é reducible a el usando espacio auxiliar logarítmico.*

E coñecido ([46]), que a clase *NC* está incluída en *PTIME* (inclusión que se cree estricta). O problema *MCV (monotone circuit value)* é *log-space complete* para *PTIME*.

Dwork, Kanellakis e Mitchell, probaron que o problema MCV [77] era transferible á unificación dun determinado grafo, levando portas do circuito a nós do grafo e as conexións a líneas Pais-Fillos no novo grafo, cun custe logarítmico en espacio. Xa que o problema MCV é *log-space complete* para *PTIME*, dedúcese que a unificación tamén o é. Esto significa que, agás que $PTIME = NC$ (o que é moi pouco probable), a unificación non se pode realizar, en xeral, en tempo logarítmico en función da lonxitude da entrada. De elo, Dwork *et al.* deduciron que a unificación era unha operación difícilmente paralelizable e probaron amais que o matching en cambio sí que se atopa na clase NC. Xa que para certos problemas o matching coincide coa unificación, esto proba que existen problemas de unificación paralelizables. Yasuura, chegou independentemente ó mesmo resultado xeral e presentou un algoritmo de unificación paralelo que corre no peor caso en $\mathcal{O}(log^2N + VlogV)$ onde N é o número de nós e V o de variables do grafo formado polo par de termos a unificar.

Vitter e Simons prantexáronse atopar un algoritmo paralelo de unificación efectivo. Deron como medida da efectividade a de atoparse nas clases PC e, especialmente, na PC^*. Como un crecimento cúbico ou superior do número de procesadores non é aceptable, Vitter e Simons aducen que o modelo escollido por Dwork *et al.* non é o máis idóneo para coñecer os algoritmos realmente paralelizables e sí que o é, en cambio, a pertenencia a PC^*. Proban que o algoritmo de Paterson e Wegman pode paralelizarse de forma que o custe en tempo veña dado por $O(E/P + VlogP)$. En base ós resultados de Yasuura e de Vitter e Simons dedúcese que para un número de variables baixo, a unificación é paralelizable. Kanellakis e Revesz [97] probaron que os problemas cun número fixo de variables, son parelalizables non só no senso de Vitter e Simons, senon tamén no de Dwork (pertenencia á clase NC).

O primeiro algoritmo paralelo de unificación foi o de Yasuura. A partir do grafo dos termos, crea un hipergrafo con dous tipos de líneas : as do grafo dado (de pais a fillos) e as que corresponden ós subtermos que se unifican conxuntamente cunha variable. Cada nivel do hipergrafo pode estar conectado a un nivel inferior por estas líneas. Só pode ser atravesado un nivel de cada vez, polo que soamente as que se atopan nun mesmo nivel poden ser procesadas en paralelo. Yasuura proba que este encadeamento de líneas pode ter unha lonxitude que non excede de $log_2 V$ con V variables do par de termos.

Vitter e Simons deron dous algoritmos de unificación paralela. O primeiro é unha versión paralela do algoritmo de Huet, mentras que o segundo éo do de Paterson e Wegman. Empregan o paralelismo para :

- crear líneas de fillos a pais,
- procesar en paralelo tódolos nós equivalentes a un dado,
- seleccionar o pai do nó actual para ser inmediatamente procesado,
- actualizar, ou crear se é necesario, os punteiros entre os nós necesariamente conectados co nó actual,
- crear as líneas entre os fillos correspondentes a nós xa conectados.

Sibai desenrolou un sistema de unificación paralela con unidades separadas de matching e datos se ben a instanciación das variables é realizada secuencialmente. Cando unha variable xa instanciada recibe posteriormente un novo valor, este e a instanciación son enviados novamente ás unidades de matching. Os nomes das variables gárdanse en memoria CAM (*content-addressable memory*), mentras que as instanciacións (datos) almacénanse en memoria RAM en forma consecutiva dependendo do número de argumentos e dos seus subargumentos. Emprégase un sinalador para coñecer o tipo de instanciación das variables (constante, variable o funcional). Cando o sinalador da instanciación dunha variable é do tipo variable, gárdase como instanciación temporal (á espera de que esta sexa instanciada). Noutro caso é unha instanciación total (e este valor é devolto como resposta), polo que as sustitucións resultantes áchanse en forma factorizada.

Existen outros desenrolos que incluen a paralelización da unificación xunto coa doutras partes (e.g. control do backtracing) na programación lóxica.

Para evitar conflictos entre os resultados de distintos procesadores (e.g. producir diferentes instanciacións para unha mesma variable), débese cuidar a forma de selección dos datos para evitar errores ou instanciacións innecesarias. Por elo, a unificación paralelízase :

- cando non existen variables comúns en varias expresións. Esto pode detectarse por un análisis dinámico ou estático dos datos[40],

- cando se combina a paralelización para a detección de conflictos e eliminación de operadores, coa secuenciación nas instanciacións das variables[166],
- no chequeo de ciclos e en operacións de preparación do algoritmo e extracción de resultados [181].

3.3 Propiedades da Unificación e Unificación de Substitucións

3.3.1 Unificación en casos particulares

En certos casos pódense utilizar algoritmos específicos de unificación. Entre estes se atopan :

- a unificación de termos lineais,
- a unificación de termos con variables separadas,
- a unificación sen test de ciclos

A unificación sen test de ciclos, tamén coñecida como *unificación racional,* permite instanciacións nas que a variable teóricamente eliminada pode atoparse na instanciación. O primeiro algoritmo onde aparece este tipo de unificación é o de Huet [85]. Tal unificación ten sido amplamente aproveitada na maioría das versións de PROLOG para evitar o alto custe do test de ciclos [42]. Curiosamente, e como observa Knight [107], o algoritmo de Paterson e Wegman non detecta a unificabilidade de *termos racionais* en tempo lineal.

O resultado da unificación racional é unha sustitución de termos en $\mathcal{T}^{\mathcal{Q}}(X)$ cuios elementos son conocidos como termos racionais (Colmerauer [42]). Estes termos poden ter lonxitude infinita pero a aridade de todo operador é necesariamente finita e só se permite un número finito de subtermos diferentes.

Exemplo 3.8 $\Gamma = \{x \stackrel{?}{=} f(x)\}$ *non son unificables, xa que para toda sustitución* σ*, a lonxitude de* $x\sigma$ *é menor que a de* $f(x)\sigma$*. En* $\mathcal{T}^{\mathcal{Q}}(X)$*, unha sustitución* σ *pode definirse recursivamente por* $x\sigma = f(x\sigma)$*. Polo tanto,* $x\sigma = f(x\sigma) = f(f(\ldots f(x\sigma)) = \ldots$*. Por simplicidade na notación, dise que a sustitución* $x\sigma = f(x)$ *é un unificador racional de* Γ*.*

Huet[85] introduce os elementos de $\mathcal{T}^{\mathcal{Q}}(X)$ como pares (V, t), onde V é un conxunto finito de variables (instanciadas conxuntamente) e t é un termo en forma clásica. Se $V \cap Var(t) = \emptyset$ trátase do termo t. Se t é un termo non variable, a clase V de variables é instanciada no termo t.

Os *termos infinitos*[47, 54], que son a sua vez extensión dos termos racionais, son tamén obxectos libres nunha categoría de Σ-álxebras denominada de *álxebras de*

contracción definidas por propiedades métricas. Diaconescu realizou unha adaptación do procedemento de Robinson para a unificación de termos infinitos.

A unificación de termos lineais cando un termo non ten variables en común co resto, é notoriamente máis sinxela que a unificación usual, xa que non é necesario intentar transmitir as instanciacións e tampoco é preciso comprobar a ausencia de ciclos.

3.3 *Sexan s,t termos tais que* $Var(s) \cap Var(t) = \emptyset$ *con s lineal e cun unificador racional* θ *Entón :*

- θ *é un unificador de* s, t,
- *a unificación de s e t, pode obterse reemplazando as regras de transformación Pegado de clases e Eliminación de variables pola regra Eliminación de variables 2 onde*
  - **Eliminación de variables 2** $\Gamma \cup \{x \stackrel{?}{=} s'\} \Rightarrow \Gamma \circ \{x/s'\} \cup \{x \stackrel{?}{=} s'\}$ *se* $x \in Var(t)$, $s' \notin V$.

  Nótese que para as ecuacións da forma $x \stackrel{?}{=} t'$, $x \in Var(s)$ *non é preciso aplicar nengunha transformación, xa que se atopan en forma resolta.*

Un resultado semellante aparece en [128], se ben alí utilízase a transmisión de a instanciación das variables do termo lineal (regra UR3).

No Apéndice A, móstrase unha implementación dun algoritmo orixinal de unificación, específico para problemas que verifican as condicións da propiedade anterior e que utiliza moitas das técnicas do algoritmo de Martelli e Montanari[119].

3.4 *Sexan* s, t, σ, *tal que* $Var(s) \cap Var(t) = \emptyset, Dom(\sigma) \subseteq Var(s) \cup Var(t), s\sigma = t\sigma$ *e se* $x \in Var(t), y \in Var(x\sigma)$ *entón ou ben* $y \in Var(s)$ *ou* $\exists z \in Var(s), y \in Var(z\sigma)$

Proba : Para toda posición $\pi \in \Pi_x(t)$ verifícase que $x\sigma = (s/\pi)\sigma$. Se existe $y \in Var(x\sigma) \cap Var(t)$, tense que $(s/\pi)\sigma \neq s/\pi$. Entón debe existir $z \in Var(s/\pi), y \in Var(z\sigma)$ □.

3.5 *Sexan s e t termos unificables e* $\delta = mgu(s,t)$ *e* $\pi \in \Pi(s) \cap \Pi(t)$. *Entón se* $\delta' = mgu(s/\pi, t/\pi)$, $\delta' \leq \delta$.

3.6 *Sexan s e t termos e* ρ *unha sustitución tal que* $Var(s) \cap (Var(t) \cup Dom(\rho) \cup I(\rho)) = \emptyset$. *Se existe* $\delta = mgu(s, t\rho)$ *entón existe* $\delta' = mgu(s,t) \leq \rho\delta$.

Proba. Pola separación das variables, $s\rho = s$, de onde $\rho\delta$ é un unificador de s e t con $mgu(s,t) \leq \rho\delta$ □.

Das propiedades da igualdade dedúcese que a unificación verifica a asociatividade e conmutatividade. Como consecuencia do teorema de Robinson (e dada a existencia

dun unificador máis xeral) dedúcese a existencia dun supremo *sup*. A parte común dun conxunto de termos, produce o ínfimo deles. Por elo, entre outras propiedades do unificador máis xeral :

3.7 ([107]) *Dados dous termos s,t existe un supremo de $[s]$ e $[t]$ en $\hat{\mathcal{T}}$. Amais :*

i) *mgu(s,sup(t,u)) = mgu(sup(s,t),u),*

i)) *mgu(s,t)= mgu(t,s)*

A obtención do supremo dun conxunto de termos, faise explícita nos procedementos que, como o de Huet[85], empregan as relacións *válidas*. Nos procedementos que empregan a regra de descomposición o supremo non aparece tan claramente, pois os correspondentes termos son reducidos. O uso da descomposición permite utilizar expresións máis sinxelas que as iniciais, pero hai algúns casos nos que é necesario conservar o supremo das expresións de partida (e.g. no campo da proba automática, onde certas expresións relevantes, son utilizadas repetidamente [186] e polo tanto é moi costoso redescubrir un mesmo resultado). Esta é, según Wos e Mac Cune[186], unha das caraterísticas que diferencian á proba automática da programación lóxica, onde xeralmente só se conserva a instanciación realizada. Neste senso, na operación de unificación é importante non só a información recollida (sustitución) como o valor que toman as descripcións igualadas (supremo).

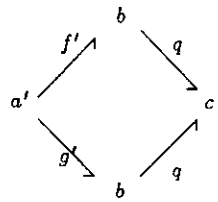
3.3.2 Unificación de substitucións.

A unificación de sustitucións ten sido utilizada entre outros por Yelick e por Rety[189, 147] na unificación ecuacional e Burstall e Rydeheard[161] probaron que se podía dar unha interpretación na lingoaxe de categorías do proceso de extración do *mgu* como unha operación de obtención de coigualadores en *Catsubst*. Rydeheard e Stell [162] mostraron como pode realizarse en *Catsubst* a unificación ecuacional.

A diferencia máis notoria entre a unificación de substitucións e as unificacións de termos e de multiecuacións, é que as variables utilizadas se atopan explicitadas. Por elo, non é necesario empregar conxuntos de protección das variables do problema e a introducción de novas variables correspóndese co coproducto dos correspondentes conxuntos de variables.

As operacións empregadas por Robinson para obter o unificador máis xeral, poden interpretarse como o resultado de aplicar teoremas de categorías. Xa que toda substitución σ pode descompoñerse na forma $\sigma_{ep} \circ j_{X'}$ onde σ_{ep} é épica, o unificador máis xeral corresponde con bastante exactitude ó coigualador de substitucións. Cando o conxunto de pares ten cardinal maior de 1, pode extraerse o unificador dun par, eliminarse as variables instanciadas en o resto dos pares e continuar o proceso. Esta é unha forma de obter coigualadores en categorías con coproductos.

3.8 ([161]) *Se $q : b \to c$ é o coigualador do par de morfismos $I \underset{g}{\overset{f}{\rightrightarrows}} b$ e $r : c \to d$ é o coigualador de*



entón $q * r : b \to d$ é o coigualador do seguinte par de morfismos :$a+a' \underset{\langle g,g'\rangle}{\overset{\langle f,f'\rangle}{\rightrightarrows}} b$ [7]

3.9 ([161]) *Para tódolos morfismos épicos $h'_a \to a$, cúmplese que $q : b \to c$ é un coigualador do par de morfismos $a \underset{g}{\overset{f}{\rightrightarrows}} b$ sse é un coigualador do par $a' \underset{g*h}{\overset{f*h}{\rightrightarrows}} b$.*

A propiedade anterior, en *Catsubst* corresponde á eliminación dos operadores superiores comúns ó par de substitucións a unificar.

3.10 ([161]) *Sexa C a clase de pares de substitucións coigualables. Existe unha función única agás isomorfismos,*

$$\phi : C \to Substitucions$$

definida a partir dos pares de substitucións irreducibles, e que satisfai as seguintes ecuacións (no senso de que se está definido o termo da dereita, tamén o está o da esquerda, e son iguais)

$$\phi(P \oplus Q) = \phi(P)\phi(Q \circ \phi(P)) \qquad onde\ (\sigma,\tau), \oplus(\sigma',\tau') = (<\sigma,\sigma'>, <\tau,\tau'>),$$

$$\phi(\gamma \circ P) = \phi(P)\ (onde\ \gamma\ é\ épica)$$

Por outra parte, $\phi(P)$ é o coigualador de $P \in C$.

A demostración de Rydeheard e Burstall desta propiedade é unha adaptación a este contexto da demostración de Robinson. Usando unha demostración semellante, Barr e Wells [15] proban que, na categoría de modelos dun *FP sketch libre* existe o coigualador de todo par de morfismos coigualables e que tal coigualador é tamén unha álxebra libre.

3.11 *Todo coigualador $\theta : X \rightharpoonup Y$ dun par de substitucións $I \underset{\tau}{\overset{\eta}{\rightrightarrows}} X$ pode ser expresado como $X' + X'' \overset{<\rho,\theta'>}{\rightharpoonup} Y$ onde $X' \overset{\rho}{\rightharpoonup} Y$ é un isomorfismo.*

Proba. Sexa $\Gamma = J \underset{\delta}{\overset{\xi}{\rightrightarrows}} Z$. Chamarase $medida(\Gamma) = (|Z|, \Sigma_{i \in I}(tam\_estr(i\sigma) + tam\_estr(i\delta), |Z|)$. Estas ternas pódense ser ordenadas totalmente a partir do orden léxicográfico e do orden dos naturais.

No caso de que no par $\{\eta, \tau\}$ non se poidan aplicar as propiedades 3.8, 3.9, tense que $I = \{i\}$ e atendendo ás partes épicas η', τ' de η, τ verifícase que :

[7] onde $*$ e $< f, f' >$ expresan a composición e a universalidade do coproducto de f, f' nesa categoría.

1. η' o τ' é isomorfismo,

2. $\eta' = \tau'$ o $cod(\eta') \cap cod(\tau') = \emptyset$.

1. é certo, xa que ó se realizar a descomposición do primeiro operador (2.51) ambos teñen algún operador, ou ben estes coinciden e nese caso a propiedade 3.9 elimínaos ou ben son diferentes e polo tanto non son unificables.

Para 2. nótese que se η' é isomorfismo e $cod(\eta') \cap cod(\tau') \neq \emptyset$ entón τ' é isomorfismo xa que noutro caso, as lonxitudes de $i\eta\theta$ e $i\tau\theta$ son necesariamente distintas.

En consecuencia, de 1. e 2. dedúcese que se η' é isomorfismo, sexa $i\eta' = x$, tense que $X = \{x\} + X''$ e $< 1_{X''}, (\eta')^{-1}\tau' >$ é un coigualador de η, τ. Pola unicidade do coigualdor existe ρ, $< 1_{X''}, (\eta')^{-1}\tau' > \rho =< \rho, (\eta')^{-1}\tau\rho >= \theta$.

Para probar o resultado, aplicarase inducción na $medida(\{\eta, \tau\})$. Dado que para os casos básicos (aqueles nos que non se pode aplicar 3.8 nen 3.9) o resultado é certo, tamén se verifica para o caso no que se poida aplicar a propiedade 3.8, dado que non modifica o coigualador e que ó eliminar un par de operadores redúcese o valor de $\Sigma_{i \in I}(tam\_estr(i\sigma) + tam\_estr(i\delta)$.

Para realizar a proba no caso de que $I = I_1 + I_2$, $|I| > 1$, empregarase a propiedade 3.8. Sexa θ_1 o coigualador de $j_{I_1} \circ \eta, j_{I_1} \circ \tau$. Por inducción, pode supoñerse que $X = X_1' + X_2''$, $\theta_1 : X \rightarrow Y_1 = < \rho_1, \theta_1' >$ xa que $medida(\{\eta, \tau\}) > medida(\{j_{I_1} \circ \eta, j_{I_1} \circ \tau\})$. Amais $medida(\{j_{I_2} \circ \eta\theta_1, j_{I_2} \circ \tau\theta_1\}) < medida(\eta, \tau)$ xa que : se θ_1 é isomorfismo, a terceira compoñente da medida redúcese sen incrementar as outras duas. Se θ_1 non é isomorfismo, $|X_1'| < |X|$. Aplicando novamente inducción obtense $Y_1 = Y_1' + Y_1''$, $\theta_2 = < \rho_2, \theta_2' >$, de onde $\theta = < \rho_1, \theta_1' > \circ < \rho_2, \theta_2' > = < \rho_1\rho_2, \rho_1\theta_2', \theta_1'\rho_2, \theta_1'\theta_2' >$, $\rho = \rho_1\rho_2$ □.

De feito tamén se probou que existen coigualadores da forma $< 1_{X'}, \sigma >$. Xa que θ é coigualador de Γ entón $\theta = < 1_{X'}, \sigma >$ $\lambda = < \lambda, \sigma\lambda >$ onde λ é isomorfismo pola unicidade do coigualador □.

Polo tanto, dentro dos coigualadores, existe unha clase que se corresponde cos *mgu* idempotentes estudiados no capítulo anterior.

3.12 *Se C é unha categoría con coproductos, todo morfismo do tipo $< \rho, \theta >$ onde ρ é un isomorfismo, é o coigualador dalgún par de morfismos.*

Proba. Sexan $X' + X'' \underset{\tau}{\overset{\eta}{\rightrightarrows}} X' + X''$ onde $\eta = 1_{X'+X''}$, $\tau = < \rho, \theta >$ $\rho^{-1} \circ j_{X'}$.

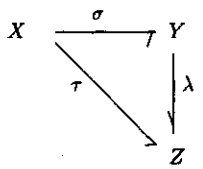
- ηo $< \rho, \theta >$ vén caracterizado por
  - $j_{X'} \circ \eta$ $< \rho, \theta > = \rho$,
  - $j_{X''} \circ \eta$ $< \rho, \theta > = \theta$,
- τ $< \rho, \theta >$ vén caracterizado por
  - $j_{X'} \circ \tau$ $< \rho, \theta > = \rho\rho^{-1} \circ j_{X'}$o $< \rho, \theta > = \rho$,

$$- \; j_{X''} \circ \tau \; <\rho,\theta> = \theta\rho^{-1} \circ j_{X'} \circ \; <\rho,\theta> = \theta\rho^{-1}\rho = \theta.$$

Sexa δ, un unificador de η e de τ. Entón $<\rho,\theta> \; \circ\rho^{-1} \circ j_{X'} \circ \delta = \delta$ e polo tanto $<\rho,\theta> \leq \delta$. Para a unicidade, sexan γ, γ' tal que $<\rho,\theta> \; \circ\gamma = <\rho,\theta> \circ\gamma'$, de onde $\rho \circ \gamma = j_{X'} \circ \; <\rho,\theta> \; \circ\gamma = j_{X'} \circ \; <\rho,\theta> \; \circ\gamma' = \rho \circ \gamma'$ e, xa que ρ é isomorfismo, $\gamma = \gamma'$ □.

Categorías de Coigualadores en *Catsubst*

Sexa X un conxunto finito. Lémbrase que a categoría co-slice $X/Catsubst$ (en adiante X/Cs) ten como obxectos as substitucións da forma $X \rightarrow Y$, e como morfismos $\sigma \rightarrow \tau$ ás substitucións λ que fan o diagrama :

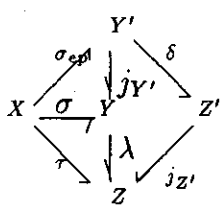


conmutativo. Esta categoría está formada por tódolos unificadores dalgún problema $\sigma, \tau : I \rightrightarrows X$, xa que toda substitución é unificador do par trivial $\Gamma = \{\tau, \tau\}$.

Esta categoría ten como subcategoría a aquela cuios obxectos son as substitucións épicas $X \rightarrow Y$. Esta subcategoría será representada por X/Cs_{ep}, e a aplicación que leva toda substitución a súa correspondente parte épica tamén é un funtor $X/Cs \rightarrow X/Cs_{ep}$.

3.13 *Entre as categorías X/Cs e X/Cs_{ep} existe un funtor G, que leva a toda substitución na parte épica da sua factorización minimal.*

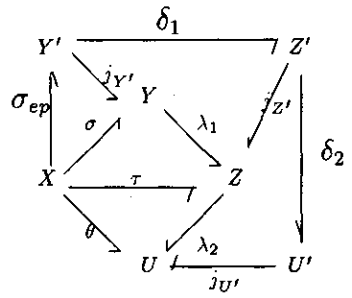
Sexa $\sigma_{ep} \circ j_{Y'}$ a factorización minimal de σ (que tamén é unha factorización epi-mono de σ). A aplicación G dada por $G(\sigma) = \sigma_{ep}$ esténdese para obter a actuación do funtor nun morfismo $\sigma \xrightarrow{\lambda} \tau$. Para elo, obsérvase que a parte épica τ_{ep} de $\tau = \sigma\lambda$ factorízase por σ_{ep}. Denotando por $\delta \circ j_{Z'}$ a factorización epi-mono de $j_{Y'} \circ \lambda$, entón $\sigma_{ep}\delta = \tau_{ep}$. A substitución $G(\lambda) = \delta$. Por ser σ_{ep} épica, δ (que tamén o é), está unívocamente determinada.



Para probar que G ten as propiedades dun funtor :

- $G(1_\sigma) = 1_{G(\sigma)}$. Coa anterior notación, $G(1_\sigma) = \delta$, onde δ é a parte épica de $j_{Y'} \circ 1_{Y'} = j_{Y'}$ que é necesariamente $1_{Y'} = 1_{G(\sigma)}$,

- $G(\lambda_1\lambda_2) = G(\lambda_1)G(\lambda_2)$. Sexan $\tau = \sigma\lambda_1$, $\theta = \tau\lambda_2$ e $\sigma_{ep} \circ j_{Y'}$, $\tau_{ep} \circ j_{Z'}$, $\theta_{ep} \circ j_{U'}$ as respectivas factorizacións minimais de σ, τ, θ respectivamente. Para obter $G(\lambda_1)$ se $\delta_1 \circ j_{Z'}$ é a factorización minimal de $j_{Y'} \circ \lambda_1$, $G(\lambda_1) = \delta_1$. Análogamente $G(\lambda_2) = \delta_2$. Polo tanto $\sigma_{ep}\delta_1 = \tau_{ep}$, $\tau_{ep}\delta_2 = \theta_{ep}$. A factorización $\delta \circ j_{U'}$ de $j_{Y'} \circ \lambda_1\lambda_2$ obtense (como no caso previo) a partir da factorización de $j_{Y'} \circ \lambda_1$, polo que $\delta = \delta_1\delta_2$ □.



Dado X, a categoría X/Cs_{ep} ten unha subcategoría de interese : aquela cuios obxectos son coigualadores dalgún problema de unificación X/Cs_{cq}. Naturalmente a identidade 1_X é coigualador dos problemas $\sigma, \sigma : X \rightrightarrows X$.

3.14 *Entre as categorías X/Cs_{ep} e X/Cs_{cq} existe un funtor F que leva cada substitución épica no coigualador do maior problema* [8]. *do que a substitución é unificador.*

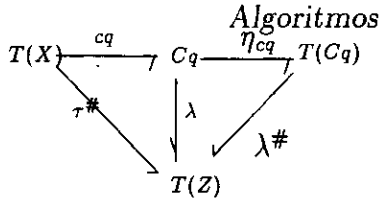
Proba. Sexa $X \xrightarrow{\sigma} Y$ unha substitución épica. En primeiro lugar obtense o parnúcleo $Pn \underset{q}{\overset{p}{\rightrightarrows}} T(X)$ (en Set) de $\sigma^\#$. Pola construcción de este en Set, Pn contén tódolos pares (t, t') que son igualados por $\sigma^\#$. Nengún destes pares póde ser da forma $t = f(\ldots), t' = g(\ldots), f \neq g$, xa que $\sigma^\#$ non os igualaría, nen tampoco o faría para $t = x, t' = f[x], t \neq x$, xa que ó aplicar $\sigma^\#$ obtense $x\sigma, t\sigma[x\sigma]$ que son necesariamente distintos por ter diferente tamaño.

Sexa $I = \{(x,t) \in Pn\}$. Este é un conxunto xerador do propio Pn, e pódese identificar Pn con $T_\Sigma(I)$, mediante $(f(\ldots,x,\ldots), f(\ldots,t,\ldots)) = f(\ldots,(x,t),\ldots)$. Pola unicidade da álxebra libre, e para simplificar a notación, non se diferenciará entre p e $p\eta_I$ e igualmente entre q e $q\eta_I$.

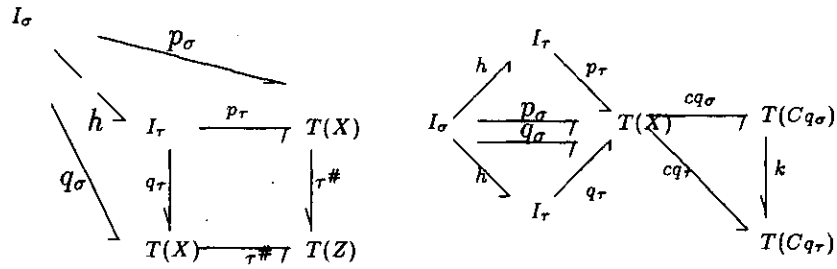
Sexa $cq : T(X) \to Cq$ o coigualador en Set de (p,q). Defínese $F(\sigma)$ como a substitución $X \rightarrow Cq$ dada por $\eta_{Cq}\, cq\, \eta_X : X \to T(Cq)$. Probarase que é un coigualador (en $Catsubst$) e que F é un funtor entre ambas categorías.

$F(\sigma)$ é o coigualador de $I \underset{q}{\overset{p}{\rightrightarrows}} X$. Para elo, sexa τ un unificador de p_σ, q_σ. Polo tanto, $\tau^\# : T(X) \to T(Z)$ coiguala $I \underset{q}{\overset{p}{\rightrightarrows}} T(X)$, de onde existe unha única aplicación $\lambda : Cq \to T(Z)$ tal que $\lambda\, cq = \tau^\#$. Así, λ correspóndese cunha substitución (de igual nome) e verifica $\lambda^\# \eta_{Cq}\, cq = \lambda\, cq = \tau^\#$. Componendo con η_X, dedúcese que $F(\sigma)\lambda = \tau$.

[8] aquel que contén o maior conxunto de pares unificados pola substitución.



Para coñecer a actuación do funtor sobre os morfismos, sexan $X \xrightarrow{\sigma} Y$, $X \xrightarrow{\tau} Z$ e unha substución $\lambda : Y \rightharpoonup Z$, $\sigma\lambda = \tau$. Polo tanto, $\lambda^{\#}\sigma^{\#} = \tau^{\#}$. Denotando por $I_\sigma \overrightarrow{\overrightarrow{p_\sigma}} T(X)$ ó par núcleo de $\sigma^{\#}$, entón $\lambda^{\#}\sigma^{\#}\, p_\sigma = \tau^{\#}\, p_\sigma = \lambda^{\#}\sigma^{\#}\, q_\sigma = \tau^{\#}\, q_\sigma$. Pola universalidade do par núcleo, existe $h : I_\sigma \to I_\tau$ tal que $p_\tau\, h = p_\sigma$, $q_\tau\, h = q_\sigma$. De onde $p_\tau\, h$, $q_\tau\, h$ son coigualados por cq_σ e pola propiedade universal do coigualador, existe un único $k : Y_\sigma \to Y_\tau$ tal que $k\, cq_\sigma = cq_\tau$. O morfismo $F(\lambda)$ en X/Cs_{cq} é xustamente $\eta_{Y_\tau}\, k$.



Para demostrar que F é un funtor, debe probarse que

a) $F(1_\sigma) = 1_{F(\sigma)}$,

b) $F(\lambda_1\lambda_2) = F(\lambda_1)F(\lambda_2)$.

a) Se se verifica que $\sigma = \tau$, a única aplicación k en $\mathcal{S}et$ que fai o diagrama conmutativo é naturalmente 1_{Cq} que corresponde a $1_{Cq} : Cq \rightharpoonup Cq$.

b) Sexan as substucións $\sigma, \tau, \theta, \lambda_1, \lambda_2$, $\sigma\lambda_1 = \tau$, $\tau\lambda_2 = \theta$.

Pola construcción anterior, e usando a mesma notación, cq_τ coiguala p_σ, q_σ, polo que existe k_1 tal que $k_1\, cq_\sigma = cq_\tau$ e análogamente k_2, k_3 con $k_2\, cq_\tau = cq_\theta$, $k_3\, cq_\sigma = cq_\theta$, de onde se obteñen substucións $\eta_{Cq_\tau}\, k_1 = F(\lambda_1)$, $\eta_{Cq_\theta}\, k_2 = F(\lambda_2)$, $F(\lambda_1\lambda_2) = \eta_{Cq_\theta}\, k_3$. Tense que $F(\lambda_1)$ é a substitución caracterizada por : $F(\sigma)F(\lambda_1) = F(\tau)$ e $F(\lambda_1\lambda_2)$ é aquela que verifica $F(\sigma)F(\lambda_1\lambda_2) = F(\theta)$. Pero, xa que $F(\sigma)F(\lambda_1)F(\lambda_2) = F(\tau)F(\lambda_2) = F(\theta)$ tense a condición buscada (obsérvese que $F(\sigma)$ é épica como consecuencia de ser coigualador). □.

Pola propiedade previa, a categoría X/Cs_{cq} ten como obxectos as substucións da forma $<\rho, \sigma>$, onde ρ é unha permutación. Amais esa proposición mostra que todo problema de unificación, ten algún coigualador da forma $<1_{X'}, \sigma>$. Estes unificadores forman a súa vez unha subcategoría de X/Cs_{cq} que será expresada por X/Cs_{idp} que equivale ás sustucións idempotentes estudiadas por Eder e por Lassez *et al.* [59, 113], se ben aquí, non só non son en xeral idempotentes, senon que amais só son idempotentes

para os problemas triviais $\sigma \stackrel{?}{=} \sigma$. Para elo, nótese que pola propiedade *2.50*, σ é idempotente sse $\sigma = <1_{X'}, \sigma'> \circ j_{X'}$. Para que sexa coigualador debe ser épica, polo que $j_{X'} = 1_X$.

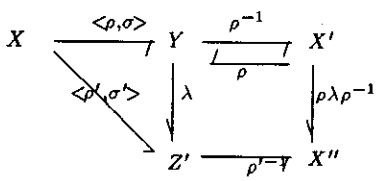
3.15 *Entre* X/Cs_{cq} *e* X/Cs_{idp} *existe unha adxunción* $H \dashv W$, *onde* $W(\sigma) = \sigma$ *e* $H(<\rho, \sigma>) = <\rho, \sigma> \circ \rho^{-1} = <1_{X'}, \sigma\rho^{-1}>$.

Proba. As condicións a verificar son :

1. que H, W son funtores,
2. para todo par de obxectos a de X/Cs_{cq} e b de X/Cs_{idp} debe existir unha bixección ϕ entre os morfismos $\lambda : a \to W(b)$ e os de $H(a) \to b$ e,
3. se $\lambda : a \to a'$, $\lambda' : a' \to W(b)$ cúmplese que $\phi(\lambda\lambda') = H(\lambda)\phi(\lambda')$ e rec´iprocamente, se $\lambda : H(a) \to b$, $\lambda' : b \to b'$, entón $\phi^{-1}(\lambda\lambda') = \phi^{-1}\lambda W(\lambda')$.

1. As actuacións de H e de W sobre os morfismos veñen dadas por : sexan $<\rho, \sigma> \stackrel{\lambda}{\to} <\rho', \sigma'>$ e $<1_{X'}, \sigma> \stackrel{\lambda'}{\to} <1_{X''}, \theta>$ morfismos respectivamente en X/Cs_{cq} e X/Cs_{idp}. Naturalmente W vén definido por $W(\lambda') = \lambda'$.

$H(\lambda) = \rho\lambda(\rho')^{-1}$ é un morfismo entre $H(<\rho, \sigma>) = <\rho, \sigma>\rho^{-1}$ e $H(<\rho', \sigma'>) = <\rho', \sigma'>(\rho')^{-1}$.



A funtorialidade obtense de :

- $H(1_{<\rho,\sigma>}) = \rho\, 1_{<\rho,\sigma>}, \rho^{-1} = \rho\rho^{-1} = 1_{H(<\rho,\sigma>)}$,
- Sexan $<\rho, \sigma> \stackrel{\lambda}{\to} <\rho', \sigma'>$, $<\rho', \sigma'> \stackrel{\lambda'}{\to} <\rho'', \sigma''>$. Tense que $H(\lambda)H(\lambda') = (\rho\lambda(\rho')^{-1}))(\rho'\lambda'(\rho'')^{-1} = \rho\lambda\rho'(\rho')^{-1}\lambda'(\rho'')^{-1}) = \rho\lambda\lambda'(\rho'')^{-1} = H(\lambda\lambda')$.

2. A unicidade de $\phi(\lambda) : <\rho, \sigma>\rho^{-1} \to <1_{X'}, \sigma'>$ é consecuencia de que $\phi(\lambda) = \rho\lambda$ verifica $(<\rho, \sigma>\rho^{-1})\rho\lambda = <1_{X'}, \sigma'>$, polo carácter épico de $<\rho, \sigma>$ (ya que é coigualador), $<\rho, \sigma>\rho^{-1}$ é épica e polo tanto $\rho\lambda$ é a única que verifica a condición.

Sexa $\lambda : H(<\rho, \sigma>) \to <1_{X''}, \theta>$. Polo tanto $<1_{X''}, \theta> = <\rho\rho^{-1}, \sigma\rho^{-1}> \circ\lambda = <\rho, \sigma> \circ\rho^{-1}\lambda$. A unicidade do morfismo $\phi^{-1}(\lambda) = \rho^{-1} \circ \lambda$ é novamente consecuencia do carácter épico de $<\rho, \sigma>$.

3. Falla probar que $\phi(\lambda\lambda') = H(\lambda)\phi(\lambda')$, onde $<\rho, \sigma> \stackrel{\lambda}{\to} <\rho', \sigma'> \stackrel{\lambda'}{\to} <1_{X''}, \sigma''>$. Esto é certo, xa que :$\phi(\lambda\lambda') = \rho(\lambda\lambda') = \rho\lambda(\rho')^{-1}\rho'\lambda' = H(\lambda)\phi(\lambda')$. A outra condición é análoga □.

A continuación móstrase a diferencia entre as substitucións que son sinxelamente coigualadores e aquelas que equivalen a sustitucións idempotentes :

Exemplo 3.9 *Sexa* $\{x,y,z\} \xrightarrow{\tau} \{y,z\}$, *dada por :* $x\tau = f(y), y\tau = z, z\tau = y$, *é coigualador xa que toma a forma* $<\rho,\sigma>$ *onde* $\rho : \{y,z\} \rightharpoonup \{y,z\}, y\rho = z, z\rho = y$ *e* $x\sigma = f(y)$. *Neste caso,* $H(\sigma) = \theta$ *con* $\{x,y,z\} \xrightarrow{\theta} \{y,z\}, x\theta = f(z), y\theta = y, z\theta = z$.

Esta adxunción é unha nova mostra de que moitas das operacións empregadas en computación proceden de propiedades destacadas das súas estructuras categóricas, que xustifican ou xeralizan tais operacións como se pode observar nos traballos de Freire *et al.* [69, 68] onde se proba a naturalidade das estructuras de control en programación, como a *Iteración, If then else*, e outras especificacións recursivas.

Capítulo 4

Unificación Ecuacional

Neste capítulo abórdase en primeiro lugar o estado actual da E-unificación. Probase que o procedemento de *narrowing* conserva a completitude se á teoría se lle incorporan os pares críticos tamén para o caso de presentacións condicionais. Expóñense os principais métodos de E-unificación, mostrando que o proceso de abstracción permite incorporar métodos coñecidos de E-unificación para a unificación ecuacional en *Catsubst*. Dase un novo método de unificación ecuacional, próbase a corrección completitude do algoritmo e incorpóranse algunhas melloras ó novo método.

4.1 Unificación Ecuacional

4.1.1 Unha visión xeral da Unificación Ecuacional.

A unificación ecuacional pode considerarse que se inicia co traballo de Plotkin[139]. Este comprendeu que a introducción de axiomas ecuacionais (sinalaba especialmente os asociativos e conmutativos) nos probadores automáticos de teoremas, supuña unha gran fonte de ineficiencia e suxiríu que podía reducirse se, para os casos máis frecuentes, se introducía un método directo para tratar con tales ecuacións. Probou que, en presencia da asociatividade, existen problemas de E-unificación con conxuntos infinitos de solucións minimais (por $\leq$) e deu unha forma de obter tódalas solucións. Atendendo ó cardinal dos conxuntos de solucións minimais, clasificou as teorías en :

- teorías unitarias : aquelas nas que todo problema de E-unificación con algunha solución, admite un E-unificador máis xeral,
- teorías finitarias : cando calquera problema de E-unificación resoluble ten algún conxunto finito de E-unificadores minimais,
- teorías infinitarias : Todo problema de E-unificación resoluble ten un conxunto (nalgúns casos infinito) de solucións minimais,

- teorías 0-arias : Son aquelas para as que existe algún problema de E-unificación sen conxuntos minimais de solucións.

A condición principal a esixir a un conxunto de E-unificadores é a completitude. Xa que o uso de axiomas fai variar o conxunto de variables do sistema, para asegurar a corrección, estas variables soen atoparse incluídas nun conxunto de variables protexidas, de forma que as novas non pertenecen ó conxunto protector. Así os E-unificadores son comprobados acerca da igualdade, exclusivamente neste conxunto. Amais, para simplificar os movementos de variables polas sustitucións, esíxese tamén a idempotencia nas solucións.

Definición 4.1 *Sexa* $U_E(s,t)$ *o conxunto de E-unificadores de* $s \stackrel{?}{=} t$, $Var(s,t) \subseteq W$. *Entón* $cU_E(s,t)$ *é un* conxunto completo de E-unificadores *de* $s \stackrel{?}{=} t$ en W, *se :*

- $cU_E(s,t) \subseteq U_E(s,t)$,
- $\forall \delta \in U_E(s,t), \exists \sigma \in cU_E(s,t), \sigma_W \leq_E \delta_W$,
- $\forall \sigma \in cU_E(s,t),\ I(\sigma) \cap (W \cup Dom(\sigma)) = \emptyset$.

$\mu U_E(s,t)$ *é un* conxunto completo minimal de E-unificadores *se amais verifícase :*

$\forall \sigma, \tau \in \mu U_E(s,t)$, *se* $\sigma_W \leq_E \tau_W$ *entón* $\sigma = \tau$.

A unificación ecuacional, móstrase aparentemente caprichosa. Nalgúns casos, engadir un axioma a unha teoría unitaria convírtea en 0-aria. Noutros sucede xustamente o contrario. Existen teorías finitas que atendendo á unificación son infinitarias.

Plotkin conxeturou a existencia de problemas resolubles sen unificadores minimais, o que foi probado por Fages e Huet[62] para un sistema de ecucións canónico. Baader [7, 10] mostrou exemplos de teorias sinxelas que tamén son 0-arias e deu unha caracterización do tipo de unificación dunha teoría se esta define unha variedade de semigrupos idempotentes, caracterización dada a partir propiedades das variedades.

As teorías que teñen tipo unitario ou finitario, xogan un importante papel na proba automática, en xeralizacións do algoritmo de Knuth-Bendix e na programación lóxica con igualdade [170, 138, 93]. Sen embargo, non existe un algoritmo xeral que, a partir dos axiomas, indique o tipo da teoría, xa que Nutt [134] probou a non decibilidade deste problema.

Incluso nas teorías finitarias, Book e Siekman [23] probaron que non son acotadas, xa que existen problemas de E-unificación cuias solucións minimais forman conxuntos finitos de cardinal arbitrario.

Bürckert *et al.* [32] probaron que existen teorías para as cales a unificación dun par de termos é infinitaria, mentras que existen sistemas sen conxuntos minimais de E-unificadores. Nun senso algo semellante, Narendran e Otto[131] atoparon unha teoría

para a que a unificación dun par é decidible non séndoo para conxuntos arbitrarios de pares.

A indecibilidade da E-unificación esténdese tamén ó matching incluso en teorías cunha presentación canónica. Bockmayr[20] encontrou, nunha especificación dos enteiros xa empregada por Rety *et al.* [148], que a decibilidade da E-unificación implicaba a decibilidade do Décimo Problema de Hilbert, cuia indecibilidade foi probada por Matiyasevich[120].

Entre outros casos de indecibilidade atópanse a asociatividade e distributividade, da asociatividade, conmutatividade e distributividade[169], asociatividade e idempotencia [6]. A distributividade (pola esquerda e pola dereita) non está aínda resolta. Kapur e Narendran[100] estableceron un gran número de resultados de decidibilidade relacionando os correspondentes problemas de unificación con problemas NP-completos como 3SAT, Mono-3SAT, uno-entre-3SAT etc. Nos seus resultados sinalan como unha das principais fontes de complexidade a non linealidade de termos nos correspondentes axiomas. Para un conxunto de axiomas E, a E-unificabilidade é semidecidible.

Inicialmente a unificación ecuacional foi estudiada en teorías particulares (como no traballo citado de Plotkin), Slagle[170] quen deu algoritmos de E-unificación en teorías con simplificadores[1], conmutatividade, asociatividade, Livesey e Siekmann[115] para a Asociativa e Conmutativa (en adiante AC) ; Siekman [167] describíu algoritmos para o caso Asociativo, Conmutativo e Idempotente.

A unificación AC ten un interese especial (e.g. permite tratar en forma eficiente a estructura dos multiconxuntos) e AC é unha teoría finitaria. Livesey e Siekmann [115] observaron a relación entre a resolución de ecuacións diofánticas e a unificación AC e deron un algoritmo para o caso dunha teoría AC cun único operador. Stickel[176] observou independentemente a mesma relación e deu un algoritmo para a unificación AC, en teorías nas que se permiten operadores libres e operadores AC, aínda que sen probar a completitude. Fages[61] deu unha sofisticada medida da complexidade, para probar que o algoritmo de Stickel é completo e finaliza. O problema da unificación AC é que, en xeral, non se obtén directamente a minimalidade. Por elo, en posteriores traballos (e.g. Fortenbacher, Lincoln e Christian [66, 114]) tense estudiado a forma de reducir o número de solucións necesarias. A solución aportada por Lincoln e Christian, ten o inconveniente de que non está probada a completitude para o caso de que existan variables repetidas.

Xa que a unificación AC soe precisar resolver ecuacións diofánticas, a rápida obtención de conxuntos minimales de solucións diofánticas, é un problema importante, dado o uso xeralizado da unificación AC en campos como a reecritura, proba automática, verificación de software etc. Huet, Lankford, Clausen e Fortenbacher, [86, 111, 41, 141] deron distintos algoritmos para a obtención de bases de solucións para as ecuacións diofánticas. Boudet, Contejean e Devie [27] propuseron un novo algoritmo onde se pospón tanto como sexa posible o reemplazamento coas solucións parciais cuestión xa sugerida en Fortenbacher[66].

[1] con regras $s \rightarrow s'$, s' subtermo de s.

teorías, adxudica non determinísticamente cada variable a unha de estas, de forma que nas ecuacións con operadores doutras teorías, esa variable é considerada como variable a eliminar. Para elo, necesítase un algoritmo de E-unificación en cada teoría por separado, onde se incorpora un número arbitrario de constantes libres, xunto cun método de eliminación de constantes. A adición de constantes complica máis do esperado algúns problemas : Bürckert e Schmidt-Schauß mostraron un exemplo no que a unificación convírtese en indecidible no caso de engadir constantes libres. Boudet, Jouannaud e Schmidt-Schauß[28] aplicaron un algoritmo de E-unificación propio para teorías combinadas que demora as instanciacións, e aplicáronno ós anelos e grupos abeliáns. Posteriores melloras do método de Schmidt-Schauß son as de Boudet e de Baader e Schultz[25, 11]. Boudet evita a identificación que é a adxudicación non determinística de teorías ás variables. Para resolver o problema $s \stackrel{?}{=} t$ onde s e t teñen os seus símbolos principais en distintas teorías, un dos dous termos debe colapsarse nunha variable. Para eliminar ciclos, ou unha das variables que transmite o ciclo é eliminada ou un termo colápsase nunha variable. Baader e Schultz retoman a identificación das variables e utilizan un orden lineal nos símbolos que permite reemplazar a existencia de eliminadores de constantes nas subteorías pola existencia de E_i-unificadores de problemas coa *restricción lineal de constantes.* Baader e Schultz amais proban que o seu método permite obter a decibilidade da E-unificación a partir da decibilidade das E_i-unificacións e aplícanno ás teorías co axioma asociativo, xa que se ben son infinitarias, a unificación asociativa é decidible.

Cando as teorías a combinar teñen operadores comúns non se poden utilizar os mecanismos descritos anteriormente. Sen embargo, Ringeissen[149] deu o primeiro algoritmo que utiliza outros de E_i-unificación para a teoría combinada cando as subteorías comparten constantes (non necesariamente libres), de forma que os símbolos das ecuacións non son disxuntos. Para elo, as constantes compartidas (que se atopan en ecuacións de ambas teorías) son renomeadas. Desta forma o novo problema corresponde á E-unificación de teorías con símbolos disxuntos. Para evitar a pérdida de solucións realízanse instanciacións non determinísticas das variables dos problemas separados nas constantes compartidas como no seguinte caso :

Exemplo 4.1 $R_1 = \{x * 0 = 0\}, R_2 = \{x + 0 = 0\}$, *o problema* $\{y \stackrel{?}{=} x + 0, y \stackrel{?}{=} x * 0\}$ *é instanciado, entre outros, en* $\{0 \stackrel{?}{=} x + 0, 0 \stackrel{?}{=} x * 0\}$, *o que permite obter a solución* $\sigma = \{y/0\}$.

Estes resultados deberán ser próximamente estendidos para outros operadores compartidos non constantes en condicións semellantes e.g. sustituir as constantes por termos concretos.

Baader utilizou o esquema de unificación proposto por Rydeheard e Burstall[160] para probar que as chamadas teorías conmutativas [2] (entre as que se atopan os monoides abeliáns, monoides abeliáns idempotentes, monoides abeliáns idempotentes

[2] aquelas cuia categoría de substitucións é semiaditiva.

Entre outras técnicas empregadas en a E-unificación que naceron ou directamente relacionadas co estudio da Unificación AC atópanse :

- a abstracción de termos,
- medidas refinadas da complexidade dun problema,
- a $(R \cup E)$-unificación con R un sistema de reescrita e para o que se supón existe un algoritmo de E-unificación,
- o emprego da regra *Descomposición* para reducir o tamaño dos termos en teorías que inclúen a AC (teorías permutativas e sintácticas),
- órdenes de simplificación para a reescrita e compatibles coas ecuacións.

Unha extensión dos resultados e métodos empregados por Stickel, permitíu a Yelick e a Tiden[188, 189, 179] obter algoritmos de E-unificación en teorías combinadas se éstas son o resultado de reunir teorías sen símbolos comúns e con algoritmos de E-unificación dados. Implícitamente, esto xa se atopaba no método de Stickel para o caso dunha teoría cun símbolo Asociativo e Conmutativo e outra teoría ben sen ecuacións, ou con símbolos Asociativos e Conmutativos. No caso de Yelick, as teorías deben ser regulares e sen ecuacións colapsadas e Tiden esixe só a condición de non ter ecuacións colapsadas. Para este último caso, Tiden emprega a noción de conxunto de eliminadores dun subtermo. Para probar a terminación, se os algoritmos das respectivas teorías a combinar rematan, Yelick utilizou unha variante da complexidade empregada por Fages. O método de unificación de teorías combinadas de Herold[82] é unha extensión do método de Livesey e Siekmann para a unificación AC e é válido para combinación de teorías que sexan finitarias, regulares e sen axiomas colapsados. Herold reemplaza os subtermos que rompen a homoxeneidade dos termos por novas constantes.

O método de obtención de algoritmos de unificación ecuacional en teorías combinadas desenrolado por Kirchner[103], utiliza o esquema de transformación de ecuacións de Martelli e Montanari. Aplica a *Descomposición* a tódalas expresións onde é posible e a continuación emprega unha forma (específica para cada teoría) de transformar un problema non descompoñible (esto é cando existen termos non variables cuio símbolo superior está nesa teoría) en novos problemas onde os primeiros símbolos teñen sido eliminados. Aplicando recursivamente este procedemento obtense un sistema de ecuacións descomposto, xa que se problema dado é resoluble unha ecuación non está formada por dous termos non variables.

Schmidt-Schauß[163, 164], desenrolou unha nova forma de obter un algoritmo de E-unificación na combinación das teorías e que reune as técnicas de Kirchner coa abstracción por variables e o uso de constantes libres. Para estender a combinación de teorías precísase un sistema de eliminación de constantes libres nunha forma que xeralice, para teorías non regulares, o método de Tiden (aplicando a eliminación de constantes). Para as variables que se atopan en problemas puros de cada unha das

con involución), ou ben son unitarias ou son 0-arias. Baader e Nutt[12] probaron que as teorías conmutativas (definidas categóricamente) e as chamadas monoidais (definidas en forma alxebraica) podían integrarse e de feito toda teoría conmutativa ten unha teoría monoidal asociada e os problemas correspondentes de E-unificación son equivalentes.

As teorías Asociativas e Conmutativas inclúense nas teorías permutativas e éstas nas finitas. Schmidth-Schauß [165] probou que as teorías permutativas tampouco son decidibles e conxeturou que as teorías de variable-permutadas son decidibles.

Bürckert, Herold e Schmidt-Schauß[32] estudiaron outros resultados de decidibilidade da E-unificación, E-matching noutras xeralizacións da Asociatividade e Conmutatividade. Rusinowitch e Vigneron[157] desenrolaron un sistema de inferencia no campo da proba automática que utiliza un algoritmo de AC-unificación o que permite evitar os axiomas reflexivos funcionais e os axiomas AC.

A tese de Kirchner[103] deu orixe a novas técnicas de xeración de algoritmos de E-unificación: unificación AC; unificación en teorías sintácticas; unificación en teorías combinadas. Estes algoritmos actúan sobre multiconxuntos de pares de termos mediante unha serie de regras de transformación, e introducindo novas regras e formas de control da súa aplicación. Kirchner probou que existe unha forma de xeralizar o algoritmo de Martelli e Montanari para obter algoritmos de E-unificación. O seu método consiste en aplicar tres tipos de regras sobre cada problema :

- a regra de *Descomposición* que elimina os operadores superiores,
- a fusión ou reunión de termos que necesariamente son instanciados conxuntamente cunha mesma variable,
- a regra de mutación que corresponde ó uso de axiomas no problema dado.

Para as teorías sintácticas, a mutación pode ser obtida dunha forma moi xeral:

Mutación $\Gamma \cup \{f(s_1,\ldots,s_n) \stackrel{?}{=} g(t_1,\ldots,t_m) \Rightarrow \Gamma \cup \{s_1 \stackrel{?}{=} u_i,\ldots,s_n \stackrel{?}{=} u_n)\} \cup \{t_1 \stackrel{?}{=} v_i,\ldots,t_m \stackrel{?}{=} t_m\}$ onde $f \neq g$ e $(f(u_1\ldots,u_n),g(v_1,\ldots,v_m)) \in E$ (para $f = g$ aplícanse tanto unha regra semellante como a *Descomposición*). Da asemade un procedemento de complección que, a partir dunha presentación arbitraria, produce, no caso da terminación, unha presentación resolvente. Deste xeito pódense implementar bibliotecas de algoritmos de E-unificación como o construído en REVEUR3 [104]

Klay [106]probou que a unificabilidade en teorías sintácticas non é decidible, asociando unha teoría ecuacional a cada instancia do *Problema de Correspondencia de Post.* Amais, dada unha presentación regular, lineal e libre de colapso dunha teoría, nen sequera é semidecidible se a presentación é sintáctica. A teoría AC cun operador, ten unha presentación resolvente con 7 axiomas. Kirchner e Klay[105] probaron que tódalas teorías de unificación finitaria son sintácticas. Entre as teorías non sintácticas, hai algunhas *teorías n-sintácticas* que teñen algoritmos específicos de E-unificación[26].

Como unha proba máis da complexidade da E-unificación, nas teorías Ω-libres (para as cales a *Descomposición* pode ser aplicada en forma determinística), danse os catro tipos de E-unificación[178].

4.1.2 Unificación Ecuacional e Narrowing

Xa foi mencionado o interese por empregar unha orientación da igualdade mediante a técnica da reescrita, especialmente se se dispón dunha presentación canónica dos axiomas. Tradicionalmente, tense asociado a esta situación unha forma de extraer unificadores a partir dunha derivación denominada *narrowing* e que foi empregada inicialmente por Slagle[170] para a unificación con simplificadores. Fay[64] probou a completitude do *narrowing* como método para obter E-unificadores en teorías canónicas. Existen distintas versións deste, polo que se aclara a notación a empregar.

Definición 4.2 *Sexa s un termo, $\pi \in \Pi_\Sigma(s)$, $l \to r$ unha variante dunha regra ecuacional. Dirase que s é levado por* narrowing *(ou tamén por* narrowing simple*) en s', o que se notará por $s \overset{\eta}{\Rightarrow}_{\pi,l\to r,\sigma} s'$, ou tamén por $s \overset{\eta}{\Rightarrow}_\sigma s'$, se :*

- *existe $\sigma = mgu(s/\pi, l)$,*
- *$s' = (s)\sigma[\pi \leftarrow (r)\sigma]$.*

O narrowing con reducción $\overset{\eta r}{\Rightarrow}$ *(narrowing en [170, 64]) substitúe a última condición por $s' = s\sigma[\pi \leftarrow r\sigma]\downarrow$.*

Hullot[91] empregou o narrowing simple cunha importante restricción : a sucesión $s_0 \overset{\eta}{\Rightarrow}_{(\pi_0,l_0\to r_0,\rho_0)} s_1 \dots s_i \overset{\eta}{\Rightarrow}_{(\pi_i,l_i\to r_i,\rho_i)} s_{i+1}$ é unha secuencia de derivacións de basic narrowing ($\overset{bn}{\Rightarrow}$) *se para todo $j, 0 \le j \le i$, $\pi_j \in B_j$, onde B_j é o* conxunto de posicións básicas *definido recursivamente por :*

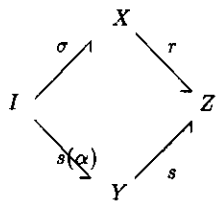
- $B_0 = \Pi_\Sigma(s_0)$,
- $B_j = B_{j-1} \backslash \{\pi \in B_{j-1}, \pi_{j-1} \le \pi\} \cup \{\pi_{j-1} \cdot \omega, \omega \in \Pi_\Sigma(r_{j-1})\}$.

As anteriores definicións exténdense a un par de termos s, t na forma : $s, t \overset{\eta}{\Rightarrow}_\sigma s', t'$ se $s \overset{\eta}{\Rightarrow}_{\pi,l\to r,\sigma} s'$, $t' = t\sigma$ ou ben $t \overset{\eta}{\Rightarrow}_{\pi,l\to r,\sigma} t'$, $s' = s\sigma$.

σ é un E-unificador de s e t obtido por narrowing, se existe unha secuencia $eq(s,t) \overset{\eta}{\Rightarrow}_{(\omega_0,\alpha_0,\sigma_0)} eq(s_1,t_1) \dots eq(s_{n-1},t_{n-1}) \overset{\eta}{\Rightarrow}_{(\omega_{n-1},\alpha_{n-1},\sigma_{n-1})} eq(s_n,t_n)$, $\mu = mgu(s_n,t_n)$, $\sigma = \sigma_0 \dots \sigma_{n-1}\mu$.

Rydeheard e Stell[162] deron unha definición de narrowing no contexto de 2-categorías :

> *Sexa $\sigma : I \to X$ unha substitución e $I \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow\alpha}} Y$ unha 2-célula e supoñendo que existan morfismos r, s tales que o seguinte cadrado é un push-out:*



Entón σ *é* narrowable *por* α, r *é o* morfismo de narrowing *e* $t(\alpha) \circ s$ *o* resultado de narrowing.

A continuación enúnciase o chamado lema do levantamento ou lema de Hullot.

4.1 ([91]) *Sexan* t *e* s *termos e* η *unha sustitución en forma normal con* $t = (s)\eta$. *Para cada derivación de reescrita*

$$t = t_0 \rightarrow_{(w_0,\alpha_0,\theta_0)} t_1 \ldots t_{n-1} \rightarrow_{(w_{n-1},\alpha_{n-1},\theta_{n-1})} t_n,$$ *existe unha secuencia de narrowing*

$$s = s_0 \overset{\eta}{\Rightarrow}_{(w_0,\alpha_0,\sigma_0)} s_1 \overset{\eta}{\Rightarrow} \ldots \overset{\eta}{\Rightarrow} s_{n-1} \overset{\eta}{\Rightarrow}_{(w_{n-1},\alpha_{n-1},\sigma_{n-1})} s_n$$

e unha secuencia de sustitucións normais $\eta_0 \ldots \eta_n$ *tal que* $\eta_0 = \eta$, $\eta_{V(s)} = \sigma_0 \ldots \sigma_{i-1}\eta_i{}_{V(s)}$ *e* $t_i = (s_i)\eta_i \, (0 \leq i \leq n)$.

O seguinte algoritmo mostra unha forma de obter E-unificadores a partir de *basic narrowing.*

Algoritmo 4.1 ([24]) *E-unificación por basic narrowing.*
Entrada : o par de termos (s,t) *a unificar e o conxunto* R *de regras ecuacionais,*
Saida : unha serie de E-unificadores Θ *de* s *e* t.

unificar (t,s,S) = / onde t e s son terminos e S e unha regra que selecciona posicions maximais */*
$E = eq(s,t)$, $B = \Pi_\Sigma(eq(s,t))\backslash\{\Lambda\}, \Theta = \varepsilon$
/ E e un termino, B un conxunto de posicions e* θ *unha sustitucion */*

BUCLE :
Se $B = \emptyset$ *enton*
$\Theta = \Theta\mu$ *e devolver* $\Theta_{V(t)\cup V(s)}$ *onde* $\mu = mgu(s',t')$, *e* $E = eq(s',t')$
Noutro caso
seleccionar mediante S unha posicion maximal π *de B*
non deterministicamente executar (1-1) (1-2) e volver a BUCLE
(1-1) $B = B\backslash\{\pi\}$
(1-2) seleccionar non deterministicamente unha variante $l \rightarrow r$ *dunha regra en R tal que todalas variables en* $Var(l,r)$ *son novas e* E/π *e* l *son unificables*
$E = (E[\pi \leftarrow r])\sigma$, $\Theta = \Theta\sigma$, $\sigma = mgu(E/\pi, l)$
$B = (B - \{\pi\}) \cup \{\pi \cdot \omega | \omega$ *posicion non variable de* $r\}$

Teñen sido empregadas diversas adaptacións do lema do levantamento de Hullot co fin de probar a completitude de distintos algoritmos de E-unificación. Versións do lema atópanse en : Yamamoto[187] para estender a completitude do narrowing a teorías débilmente canónicas; Jia-Huai You[190] para obter secuencias de narrowing equivalentes a derivacións en sistemas canónicos basados en constructores ; Giovanneti e Moiso [75] para a completitude do *narrowing condicional*; Kirchner [103] para probar que, en sistemas (R, E) de rescritura canónicos módulo ecuacional, o *narrowing módulo ecuacional* produce sistemas completos de $R \cup E$-unificadores [3]; Gallier e Snyder [72] para probar a completitude do narrowing usando ecuacións e regras orientables obtidas mediante a complección por pares críticos. Tamén Bockmayr[21], Hölldobler[83], Herold[82], Bockmayr, Krischer e Werner[22], Chabin e Rety [37] utilizan adaptacións deste lema para erguer secuencias de narrowing a partir de derivacións.

Comparando *narrowing con reducción* e *basic narrowing*, obsérvase que nengún deles está estrictamente contido no outro.

Exemplo 4.2 $R = \{g(y) \to h(f(y)),\ h(x) \to k(x),\ f(f(x)) \to x\}$. *A seguinte secuencia de* basic narrowing *para* $s = g(f(x))$ *non se obtén por* narrowing con reducción *:* $g(f(x)) \overset{bn}{\Rightarrow}_{y/fx} h(f(f(x)) \overset{bn}{\Rightarrow}_{x'/ffx} k(f(f(x)))$.

Recíprocamente, existen secuencias de *narrowing con reducción* que non son posibles por *basic narrowing* :

Exemplo 4.3 $R = \{f(x) \to g(x),\ h(a) \to b\},\ f(h(x)) \overset{nr}{\Rightarrow}_{x'/hx} g(h(x)) \overset{nr}{\Rightarrow}_{x/a} g(b)$.

Rety[147] estudiou o problema de atopar un sistema completo que inclúa a *basic narrowing* e *narrowing con reducción.* Podía esperarse a completitude das dúas estratexias conxuntamente, o que non é certo como se observa no seguinte caso :

Exemplo 4.4 $R = \{g(x,y) \to f(h(x),h(y)),\ f(x,x) \to x,\ h(h(x)) \to x\}$, $\Gamma = \{g(x,x) \overset{?}{=} 0\}$. *A secuencia de* basic narrowing *:*
$g(x,x) \overset{bn}{\Rightarrow}_{x/x,y/x} f(h(x),h(x)) \overset{bn}{\Rightarrow}_{x/hy} f(y,h(h(y))) \overset{bn}{\Rightarrow}_{x''/y} f(y,y) \overset{bn}{\Rightarrow} y$
non é sustituible por nengunha secuencia de basic narrowing *con reducción, xa que empregando* $\overset{nr}{\Rightarrow}$ *no paso* $g(x,x) \overset{nr}{\Rightarrow} h(x)$, *(obtenible por dous pasos de* $\overset{bn}{\Rightarrow}$*), o termo* $h(x)$ *é a instanciación dunha variable da regra* $g(x,y) \to \ldots$ *e polo tanto de* $h(x)$ *non se deriva nengunha secuencia de basic narrowing.*

Rety aplicou o test SL (*sufficient largeness*) para eliminar algunhas destas derivacións : este test comproba se o conxunto de posicións non básicas ten algunha posición reducible, e nese caso non se acepta tal secuencia. Esta nova condición restrinxe algunhas das aplicacións de *basic narrowing* sen perder a completitude. Tamén é posible mellorar *narrowing con reducción* empregando ese test e eliminando

[3] a definición de narrowing módulo E, obtense de reemplazar $\rho = mgu(s/\pi, l)$ por $\rho \in cU_E(s/\pi, l)$.

algunhas posicións non básicas. Para elo, nun paso de narrowing simple, transmítense as posicións básicas, mentras que nun paso de reducción transmítense as novas posicións cuios antecedentes son todos básicos, chamadas posicións débilmente básicas. Rety compara ambos métodos e baixo certas condicións *SL- basic N-narrowing* está incluído en *SL-basic S-narrowing.*

Herold[82] probou que, logo dun paso de narrowing realizado na posición π, pódense descartar as aplicacións de narrowing en posicións independentes á esquerda de π, o que se denomina *left-to-right narrowing.* Krischer e Bockmayr incorporaron novos test para eliminar derivacións innecesarias, observando a situación dos subtermos despois da instanciación completa, esto é, a composición dos *mgus* obtidos en cada paso. Igual que fai Rety, empregan un paso de narrowing simple (onde só se transmiten as posicións básicas), seguido dun paso de normalización (onde se transmiten tamén as posicións débilmente básicas).

Para que o último paso da secuencia $(s_0, B_0) \stackrel{\pi}{\Rightarrow} (s_1', B_1') \rightarrow^* (s_1, WB_1) \stackrel{\pi}{\Rightarrow} \ldots (s_{i-1}, WB_{i-1}) \stackrel{\pi}{\Rightarrow} (s_i', B_i) \rightarrow^* (s_i, WB_i)$ sexa admitido, debe verificar unha serie de condicións, para o que se utiliza a composición das instanciacións locais que, tras ser aplicada a cada un dos termos previos debe verificarse :

- as posicións descartadas en B_i' non poden ser reducibles (*SL-test*),
- os termos que quedaron no interior de cada paso de narrowing, están na forma normal(*Sub test*),
- non se poden empregar nos subtermos de aplicación previa de regras outras regras de orden inferior[4] á utilizada (*Epsilon test*),
- nen poden reducirse as posicións descartadas en B_j' (*L-reduction test*).

Nun traballo posterior, Bockmayr *et al.*[22] proban que tal estratexia está incluída en *basic narrowing* e polo tanto, só proporciona solucións normalizadas. Amais, nunca unha mesma sustitución será obtida mediante distintas derivacións. Evidentemente elo non significa que proporcione solucións redundantes no senso de minimalidade xa que existen teorías canónicas 0-arias. Tamén estes *test* poden ser aplicados empregando *narrowing con reducción* conservando a completitude e o *normalizing LSE narrowing* está contido en *normalizing left-to-right basic narrowing*, que é unha variante de *left-to-right basic narrowing*, onde ó igual que en Rety[147], admítense posicións débilmente básicas (aquelas nas que tódolos antecedentes se atopan como posicións básicas).

A finalidade do *narrowing dirixido por grafos* de Chabin e Rety[37] é eliminar certas derivacións que non producen solucións por non poder obterse unha derivación completa entre os termos dun problema. Unha primeira aproximación foi desenrolada por Dershowitz e Sivakumar[53] para os operadores superiores. Chabin e Rety, constrúen un grafo onde os nós son os subtermos dos problemas e dos lados das regras.

[4]dan un orden arbitrario para as regras.

As líneas créanse tanto entre pais e fillos como entre os termos á esquerda e á dereita dunha regra e entre termos unificables. Precísase tamén unha forma de propagación das líneas. Unha vez construido o grafo (finito), para o problema $s \stackrel{?}{=} t$ só se poden utilizar aquelas secuencias de narrowing de regras das que os lados están conectados con s e t. Evidentemente, se s e t nono están, o problema non ten solución.

Outro tipo de variantes de *narrowing* son as que empregan algún tipo de selección para as posicións nas que aplicar as regras. Neste tipo atopanse o *outermost narrowing* e o *innermost narrowing.* Na primeira, do conxunto de posicións nas que unha regra pode ser efectivamente aplicada, só se seleccionan as posicións minimais. Na estratexia *innermost*, polo contrario, de tódalas posicións entre as que é posible aplicar *narrowing*, só se seleccionan as maximais. A estratexia *innermost* está directamente relacionada con *basic narrowing:* se a presentación non é ambigua unha estratexia *innermost* é tamén *basic narrowing* (*basic narrowing* non é necesariamente *innermost*).

As estratexias *outermost* corresponden a unha execución por nome, mentras que as *innermost* a unha execución por valor. Na E-unificación en teorías canónicas, nengunha de elas é completa :

Exemplo 4.5 $R = \{f(a) \rightarrow c,\ g(f(b)) \rightarrow g(c)\}$ *é canónica. No problema* $\Gamma = \{g(f(x)) \stackrel{?}{=} g(c)\}$, *obtense a solución* $\{x/a\}$ *aplicando a estratexia innermost, mentras que aplicando a outermost, a solución que se obtén é* $\{x/b\}$. *Naturalmente as dúas solucións non son equivalentes.*

Exemplo 4.6 *[58]* $R = \{f(0,0) \rightarrow 0, f(s(x),0) \rightarrow s(0), f(x,s(y)) \rightarrow s(s(0))\}$, $\Gamma = \{f(f(u,v,w) \stackrel{?}{=} 0\}$,
Por outermost narrowing non se obteñen solucións se ben o problema é resoluble dado que $\{u/0, v/0, w/0\}$ *é unha das solucións de* Γ.

Unha ventaxa notable da selección *innermost* é que produce solucións xa normalizadas.

Cando se fai necesario empregar unha ou outra estratexia a completitude só se logra impondo novas condicións á teoría. Aínda que se perde en xeralidade, evítase o indeterminismo xeral do *narrowing*.

Nunha teoría canónica, non é relevante a derivación seleccionada para lograr a forma normal dun termo. Esta forma de indeterminación é coñecida como *indeterminismo sen importar.* Na Unificación Ecuacional, a aplicación das regras sí que pode dar lugar a distintas solucións dependendo da posición e das regras empregadas o que soe expresarse (e.g. Nutt *et al.* [135]) como *indeterminismo sen coñecer.* Como se observa, téñense empregado moi diversas técnicas para reducir esta *indeterminación sen coñecer.* Agrupamos as distintas técnicas en :

- aquelas que limitan o crecimento da árbore de derivacións de narrowing mediante a imposición de restriccións ás posicións e ás regras nas que se aplica un paso de narrowing. Estas pódense dividir en :

  - as que prohiben a aplicación de regras nunha posición dada, por comparación con outras posicións nas que se pode aplicar algunha regra. Entre éstas se atopan as estratexias : *outermost* versus *innermost*, *left-to-right* e *right-to-left*,
  - aquelas outras nas que se atende á instanciación das regras ou dos termos: en *narrowing con reducción*, teñen absoluta prioridade aquelas que non realizan instanciación dos termos. No *basic narrowing* as instanciacións de variables das regras son marcadas e non se produce aplicación de novas regras en tales posicións.

- formas de podar a árbore de derivacións (eliminan follas xa creadas). Entre estas se atopan os *SL-test, L-sub test, L-epsilon test* e *L-reduction test.* Faise preciso coñecer cal ten sido a instanciación realizada para saber se esa póla xa se atopa repetida,

- aquelas que cortan as derivacións inútiles por posuír algún tipo de información previa acerca do resultado final, coñecendo que conduce a un bucle ou un fallo. O *narrowing dirixido por grafos* de Chabin e Rety e as reglas de simplificación de Rety *et al.*[148] son deste tipo.

Lembramos que o lema de levantamento de Hullot esixe que o unificador sexa unha substitución normalizada. Na completitude de *basic narrowing* utilízase, polo tanto, o feito de que o conxunto de solucións normalizadas é un conxunto completo de E-unificadores. Esto verifícase no caso de teorías canónicas. Pero é coñecido que noutras situacións ésta diferencia marca a fronteira entre a completitude e a non completitude.

Yamamoto probou a completitude de narrowing en teorías débilmente canónicas[5], así como a completitude de *basic narrowing* en teorías débilmente canónicas se son normalizables por reducción *innermost*. Yamamoto conxeturou que ésta última condición está incluída na debilidade canónica e polo tanto que *basic narrowing* é completo tamén neste caso. Recentemente Middeldorp e Hamoen [126] probaron a falsedade desta conxetura :

Exemplo 4.7 $R = \{f(x) \rightarrow g(x,x),\ a \rightarrow b,\ g(a,b) \rightarrow c,\ g(b,b) \rightarrow f(a)\}$ *e* $\Gamma = \{f(a) \stackrel{?}{=} c\}$. *Este problema de E-unificación non ten solución por* basic narrowing *pese a que* $f(a) =_{R \cup R^{-1}} c$.

Concretamente, *basic narrowing* non é tampouco completo en sistemas confluentes nen siquera respecto das solucións normalizadas. Para asegurá-la completitude, Middeldorp e Hamoen introducen as condicións da linealidade pola esquerda e non ambigüedade[6]. No caso de ter a confluencia e linealidade pola dereita, *basic narrowing* é completo respecto do conxunto de solucións normalizables[126].

[5] aquelas para as que todo termo tén unha única forma normal

[6] unha presentación é ambigua se existen pares críticos entre regras.

Para conseguir a completitude da estratexia *outermost*, Jia-Huai You esixe a linealidade pola esquerda, a non ambigüedade e a condición chamada *disciplina de constructor*, que obriga a que os *operadores definidos*[7] non se encontren no interior do lado dereito de nengunha regra. Nutt *et al.*[135] dan novas especializacións de narrowing no caso de sistemas de reescrita libres : aqueles para os que os símbolos definidos non están presentes na parte dereita de nengún termo en forma normal.

Un dos principais problemas de *narrowing* é o de que en certas teorías, aínda sendo canónicas, poden producirse infinitas secuencias de narrowing. Téñense desenrolado certas estratexias para evitar secuencias infinitas, se ben se se esixe a completitude, o resultado non sempre é satisfactorio. Rety *et al.* e Kirchner [148, 103], mostran unha forma de cortar secuencias non finitarias de narrowing que producen pólas racionais[8], que se pode resolver mediante unha formulación recursiva do conxunto de solucións :

Exemplo 4.8 *Sexa* $R = \{g(f(x,y)) \rightarrow g(y)\}$, $\Gamma = \{g(x) \stackrel{?}{=} g(0)\}$. *As derivacións de narrowing teñen unha póla racional, xa que a aplicación de narrowing* $g(x) \stackrel{?}{=} g(0) \stackrel{n}{\Rightarrow}_{(x)\rho=f(u,v)} g(v) \stackrel{?}{=} g(0)$ *pode aplicarse igualmente ó resultado. O conxunto* Δ *definido recursivamente por :*
$\Delta = \{x/0\} \cup \{\sigma$ *tal que existe* $\delta \in \Delta, x\sigma = f(y, x\delta)\}$ *é un conxunto completo de E-unificadores.*

O *narrowing dirixido por grafos* de Chabin e Rety permite detectar a inconsistencia dun problema e polo tanto eliminar moitas derivacións (algunhas non finitas) de narrowing cando non producen solucións. Entre as diversas técnicas para comprobar a inconsistencia se atopan :

- a creación de grafos (e.g. Alpuente *et al.* [3]) para a eliminación de bucles ou mencionado de Chabin e Rety, ou conexiones axeitadas (Bläsius e Siekman [19]),
- o emprego previo de regras de reescrita máis fortes que a igualdade ecuacional (para asegurá-la finitariedade e completitude), aínda a costa de que algunhas expresións non unificables pasen o correspondente test [4],
- a introducción de regras de Fallo empregadas ambiciosamente, para cortar o antes posible as derivacións inútiles [135, 83].

Os primeiros métodos empregados con este fin, poden considerarse bastante sinxelos: e.g. a separación dos operadores en *definidos* e *constructores.* Cando os operadores superiores de dous termos son constructores distintos, non existe solución[83]. En casos máis complexos, emprégase información doutro tipo, como a proporcionada por determinados grafos que informan de que o problema dado aínda sendo localmente resoluble con regras ecuacionais, xa foi detectada a imposibilidade de reunir as distintas expresións para transformar o par inicial noutro común [37].

[7] son os operadores superiores do lado esquerdo das regras.

[8] unha árbore de derivación é racional se ten unha rama propia coa que coincide en cuanto á regra a aplicar e a posición na que se aplica.

Narrowing Condicional

Un dos principais campos cara os que se dirixe a unificación ecuacional é o da programación lóxica con igualdade. Jaffar *et al.* [93] incorporan ecuacións da forma $e \leftarrow e_1, e_2, \ldots, e_m$ (onde $e_1, \ldots, e_m$ son ecuacións) ós programas lóxicos (lóxicos ecuacionais en Hölldobler[83]). Estes programas veñen dados por pares (P, E), onde P contén as cláusulas lóxicas e E é un conxunto de cláusulas ecuacionais. Xa se mencionou que o mecanismo de narrowing condicional ten sido utilizado como a base operacional das lingoaxes que incorporan ambos tipos de programas[146].

Na E-unificación de regras condicionais non existe, nen sequera nas teorías canónicas, unha correspondencia tan directa entre reescrita e narrowing. Como se observará, a existencia de variables extra fai perder a propiedade de completitude do narrowing condicional.

Definición 4.3 *Sexan : s un término, $e_1 \stackrel{?}{=} e'_1, \ldots, e_n \stackrel{?}{=} e'_n \Rightarrow l \rightarrow r$ unha regra condicional. Se dirá que s é levado por* narrowing condicional *en s', o que será expresado por $s \stackrel{nc}{\Rightarrow}_\eta s'$, se :*

- *existen π, σ, $\pi \in \Pi_\Sigma(s)$, $\sigma = mgu(s/\pi, l)$,*
- *existen sustitucións $\lambda_1, \ldots, \lambda_n$, $\lambda'_1, \ldots, \lambda'_n$, $\eta_0, \ldots \eta_n, \eta$ e termos $u_1, \ldots, u_n$, $v_1, \ldots, v_n$ tales que :*
  $\eta_0 = \sigma$ e para todo i, $1 \leq i \leq n$, tense que $e_i\eta_i \stackrel{nc}{\Rightarrow}_{\lambda_i} u_i$, $e'_i\eta_{i-1}\lambda_i \stackrel{nc}{\Rightarrow}_{\lambda'_i} v_i$,
  $\eta_i = \eta_{i-1}\lambda_i\lambda'_i(mgu(u_i\lambda'_i, v_i))$, $\eta = \eta_n$,
- *$s' = (s)\eta[\pi \leftarrow (r)\eta]$.*

En forma semellante ó caso sen condicións, obtense o narrowing aplicado a un par $s \stackrel{?}{=} t$, o que ten sido expresado asemade por $eq(s,t) \stackrel{n}{\Rightarrow} eq(s',t')$ ....[148].

Dise que S é levado por narrowing condicional demorado *en S' usando a regra condicional $e_1 \stackrel{?}{=} e'_1, \ldots, e_n \stackrel{?}{=} e'_n \Rightarrow l \rightarrow r$, o que será expresado por $S \stackrel{ncd}{\Rightarrow}_\sigma S'$ se :*

- *existen $s \stackrel{?}{=} t \in S, \pi, \sigma$, $\pi \in \Pi_\Sigma(s)$, $\sigma = mgu(s/\pi, l)$,*
- *$S' = \left(S \setminus \{s \stackrel{?}{=} t\}\right) \bigcup_{1 \leq i \leq n} \{e_i\sigma \stackrel{?}{=} e'_i\sigma\} \cup \{s\sigma[\pi \leftarrow r\sigma] \stackrel{?}{=} t\sigma\}$*

A forma de obter E-unificadores por narrowing condicional e por narrowing condicional demorado, é semellante á empregada para o caso de ecuacións sen condicións (aplicación de narrowing ata que se fagan unificables sintácticamente). Kaplan[98] probou a completitude do narrowing para o caso canónico cando non existen variables extra. Giovanetti e Moiso[75] probaron a completitude do *narrowing condicional* no caso de existencia de variables extra se a teoría é completa por niveles, empregando novamente unha versión do lema de Hullot. Deron novos resultados de completitude para distintas teorías e estratexias de narrowing.

A existencia de variables extra é a causa de diversos problemas : o primeiro é que nalgúns casos, por narrowing condicional se obteñen solucións non normalizadas, pero non as correspondentes normalizadas.

Exemplo 4.9 $R = \{h(y,x) \stackrel{?}{=} h(x,y) \stackrel{\alpha}{\Rightarrow} f(x,a) \to g(x)\ ,\ \stackrel{\beta}{\Rightarrow} h(b,a) \to h(a,b)\ ,\ \stackrel{\gamma}{\Rightarrow} a \to c\}$ *e* $\Gamma = \{f(x,x) \stackrel{?}{=} g(x)\}$. *A solución non normalizada,* $\{x/a\}$ *obtense logo de aplicar* α *e* β*, mentras que a solución normalizada* $\{x/c\}$ *non se alcanza mediante narrowing.*

A continuación móstrase cómo nalgúns casos, nen siquera se obteñen solucións por narrowing condicional.

Exemplo 4.10 $R = \{\Rightarrow a \to b,\ \Rightarrow a \to c,\ f(b,x) \stackrel{?}{=} f(x,c) \Rightarrow b \to c\}$ *e* $\Gamma = \{f(b,x) \stackrel{?}{=} f(x,c)\}$. *A aplicación de narrowing demorado da orixe a unha secuencia non finitaria de problemas equivalentes* $\{f(b,x') \stackrel{?}{=} f(x',c), \ldots, f(b,x^n) \stackrel{?}{=} f(x^n,c), \ldots$. *Sen embargo, narrowing condicional, nen siquera pode ser aplicado en* Γ.

Neste último exemplo, o problema xurde de que a regra que proporciona a confluencia de $a \to b$, $a \to c$ (que é a única regra que pode ser aplicada en $f(b,x) \stackrel{?}{=} f(x,c)$) é novamente a única que se pode aplicar na condición $f(b,x) \stackrel{?}{=} f(x,c)$. Sen embargo, dada a substitución x/a, esta regra non é necesaria para probar à igualdade $f(b,a) =_E f(a,c)$. Este problema é evitado por Giovanetti e Moiso ó impoñer a completitude por niveles.

Aínda na completitude por niveles, se os problemas e condicións se resolven esixindo solucións normalizadas, pérdese a completitude.

Exemplo 4.11 $R = \{g(x,y) \stackrel{?}{=} h(x,a) \stackrel{\alpha}{\Rightarrow} f(a) \to b,\ \stackrel{\beta}{\Rightarrow} g(f(y),y) \to h(f(y),y)\}$, $g(x,y) \stackrel{?}{=} h(x,a) \stackrel{\gamma}{\Rightarrow} g(b,a) \to h(f(a),a)\}$, $\Gamma = \{g(x,y) \stackrel{?}{=} h(x,a)\}$.
Aplicando β *obtense* $\sigma = \{x/f(y)\}$ *e o problema* $h(f(y),y) \stackrel{?}{=} h(f(y),a)$ *que se resolve por* $\{y/a\}$. *A solución resultante* $\{x/f(a), e/a\}$ *non está normalizada.*
No caso de aplicar γ *obtense como mgu* $= \{x/b, y/a\}$*, pero a condición a resolver é* $g(x',y') \stackrel{?}{=} h(x',a)$ *equivalente ó problema inicial.*

Bockmayr [21] utiliza o narrowing condicional para os sistemas de reescrita módulo ecuacional (R,E) e proba que, no caso de inexistencia de variables extra, o procedemento de *narrowing módulo ecuacional* [91, 103] pode estenderse ó *narrowing condicional módulo ecuacional* e unha adaptación do lema do levantamento é empregada para probar a completitude. Para elo utiliza a *relación de reescrita sen avaliación da premisa.* Nesta relación, a instanciación do sistema producese por *matching (E-matching)* e as condicións non producen inmediatamente instanciacións. Se un problema se reescribe nunha forma resolta, ésta tamén se obtén mediante a *reescrita sen avaliación da premisa* e recíprocamente. A relación que se establece

entre a reescrita sen avaliación da premisa e o narrowing condicional sen avaliación da premisa é semellante á existente entre a reescrita módulo ecuacional e o narrowing condicional módulo ecuacional. Unha construcción algo semellante xa foi usada por Jia-Huai You[190], o cal emprega a derivación *outermost* para probar que, nun sistema de reescrita *basado en constructores*, a estratexia *outer narrowing* é completa.

A continuación resúmense os principais resultados acerca da completitude de *narrowing, narrowing condicional, basic narrowing* e *basic narrowing condicional* [75, 126].

- Sistemas de reescrita canónicos sen variables extra :
  - narrowing condicional é completo,
  - narrowing condicional cunha función selección, que inclúe o caso da estratexia left-to-right, é completo,
  - basic narrowing non é completo (contraexemplo mostrado por Middeldorp e Hamoen [126]).
- Sistemas de reescrita completos por niveles :
  - narrowing condicional é completo,
  - basic narrowing é completo,
  - narrowing condicional non é completo respecto das solucións normalizables.

4.2 E-unificación de pares de substitucións

A continuación realízase a unificación ecuacional de pares de substitucións no caso de teorías canónicas e de teorías cunha presentación pechada. Para elo emprégase o esquema de Rydeheard e Stell[162] onde se proba a completitude dunha forma de narrowing. A diferencia de éstos autores e, como no caso xeral do narrowing, só se permite o emprego de 2-células en posicións non variables. Amais mostrarase cómo se pode incorporar a igualdade condicional.

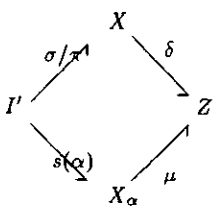
Definición 4.4 *Sexa* $\rightarrow$ *unha relación entre substitucións e pares de substitucións. Por comodidade na escritura, escribirase* $\sigma \rightarrow_\rho \sigma'$ *se se verifica que* $\sigma \rightarrow (\sigma', \rho)$. *Igualmente escribirase* $\sigma, \tau \rightarrow_\rho \sigma', \tau'$ *se* $\sigma \rightarrow_\rho \sigma'$, $\tau' = \tau\rho$ *ou ben se* $\tau \rightarrow_\rho \tau'$, $\sigma' = \sigma\rho$. *As solucións do problema* $I \overset{\sigma}{\vdash} X$, *obtidas por* $\rightarrow$ *defínense por :*

$$S(\sigma \overset{?}{=} \tau, \rightarrow) = MGU(\sigma, \tau) \bigcup_{\sigma,\tau \rightarrow_\rho \sigma',\tau'} j_X \circ \rho \circ S(\sigma' \overset{?}{=} \tau', \rightarrow)$$

onde $MGU(\sigma, \tau) = \emptyset$ *se o par non é* $\emptyset$*-unificable e* $\{\delta\}$ *se* δ *é un unificador máis xeral de* $\sigma \overset{?}{=} \tau$, *no caso de que exista e* $X \overset{j_X}{\rightarrow} X + X' \overset{\rho}{\rightarrow} Z$.

A seguinte definición, proporciona unha forma de aplicar narrowing a substitucións usando 2-células sen condicións.

Definición 4.5 *Sexan θ, σ substitucións. Dise que θ se obtén por* narrowing *dende σ, expresado por $I \overset{\alpha}{\rightarrow} X \overset{n}{\Rightarrow}_{\delta} I \overset{\theta}{\rightarrow} Z$, se existe unha 2-célula $I' \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow}} X_\alpha$ da presentación ecuacional R e unha posición π de σ, tal que :*

1. *existen substitucións δ, μ de forma que o cadrado*  *é un*

   push-out. Esta operación corresponde :

   - *a composición de σ/π e $s(\alpha)$ respectivamente, coas inclusións $X \overset{j_X}{\rightarrow} X + X_\alpha$, $X_\alpha \overset{j_{X_\alpha}}{\rightarrow} X + X_\alpha$,*
   - *a obtención de λ, coigualador de $\sigma/\pi \circ j_X \overset{?}{=} s(\alpha) \circ j_{X_\alpha}$, de onde $\delta = j_X \circ \lambda$, $\mu = j_{X_\alpha} \circ \lambda$,*

2. *a parte épica da factorización minimal de σ/π non é isomorfismo (equivale á restricción a posicións non variables),*

3. *$\theta = \sigma\delta[\pi \leftarrow t(\alpha)\mu]$.*

A relación $\overset{n}{\Rightarrow}$ pódese expresar tamén en función da 2-célula $ex(\alpha)$, (extensión de α). Neste caso, debe notarse que $ex(\alpha)$ verifica que tódalas substitucións da forma $s(ex(\alpha))/\omega$, e $t(ex(\alpha)/\omega)$ coinciden con σ/ω (agás da correspondente inclusión), xa que $cod(s(ex(\alpha))) = X + X_\alpha$.

A continuación danse dúas formas de aplicar narrowing no caso de 2-células condicionais, na primeira ($\overset{nc}{\Rightarrow}$) as condicións son inmediatamente resoltas, mentras que na segunda a resolución de éstas é demorada ($\overset{ncd}{\Rightarrow}$).

Definición 4.6 *Sexan θ, σ substitucións, R unha presentación ecuacional. Dirase que θ se obtén por* narrowing condicional *dende σ, expresado por $I \overset{\alpha}{\rightarrow} X \overset{nc}{\Rightarrow}_{\rho} I \overset{\theta}{\rightarrow} U$ se existe unha 2-célula condicional $\left(I' \overset{s(\alpha)}{\underset{t(\alpha)}{\Downarrow}} X_\alpha,\ I_\alpha \overset{c}{\underset{c'_\alpha}{\rightrightarrows}} X_\alpha + X^e_\alpha\right)$ de R e unha posición π de σ, tal que :*

1. *e 2. son as mesmas condicións que no caso anterior,*

3. *existe unha substitución $\kappa \in S(c_\alpha \circ < \mu \circ j_Z, 1_{X^e_\alpha} > \overset{?}{=} c'_\alpha \circ < \mu \circ j_Z, 1_{X^e_\alpha} >, \overset{nc}{\Rightarrow})$,*

4. *$\theta = \sigma\delta \circ j_Z \circ \kappa[\pi \leftarrow t(\alpha)\mu \circ j_Z \circ \kappa]$,*

5. $\rho = \delta \circ j_Z \circ \kappa$, $U = cod(\kappa)$.

Definición 4.7 *O par de substitucións* $I \overset{\sigma}{\underset{\tau}{\rightrightarrows}} X$ *é levado por narrowing condicional demorado en* $\sigma', \tau' : I'' \rightrightarrows Z + X^e_\alpha$ *mediante a 2-célula condicional* $\left(I' \overset{s(\alpha)}{\underset{t(\alpha)}{\rightrightarrows}} X_\alpha,\ I^c_\alpha \overset{c_\alpha}{\underset{c'_\alpha}{\rightrightarrows}} X_\alpha + X^e_\alpha\right)$ *se se verifican :*

1. *e* 2. *son as mesmas condicións que as de* $\overset{n}{\Rightarrow}$,

3. $\sigma' = <\sigma'_1, \sigma'_2>$. $\tau' = <\tau_1, \tau_2>$, *onde*

   - $\sigma'_1 = \sigma\delta \circ j_Z \circ [\pi \leftarrow t(\alpha)\mu \circ j_Z]$,
   - $\sigma'_2 = c_\alpha \circ <\mu, 1_{X^e_\alpha}>$,
   - $\tau'_1 = \tau\delta \circ j_Z$,
   - $\tau'_2 = c'_\alpha \circ <\mu, 1_{X^e_\alpha}>$.

A continuación empréganse as relacións $\succ$ que permiten dirixir a igualdade, para obter a completitude do narrowing condicional. Esta propiedade do narrowing será usada posteriormente para obter a completitude de novos algoritmos de unificación ecuacional.

Definición 4.8 *Sexa* R *un conxunto pechado de 2-células condicionais dirixido pola relación* $\succ$ *que se supón completa por niveles. O* n-grado *de* σ *é :*

- 0 *se non existe nengunha 2-célula* α *de* R *de profundidade* $\leq n$ *dirixida por* $\succ$ *(esto é tal que* $s(\alpha) \succ_n t(\alpha)$*) onde* $s(\alpha) = \sigma$,
- m *se* m *é a maior lonxitude dunha 2-célula* α *de* R *dirixida por* $\succ$ *e propia tal que* $s(\alpha) = \sigma$. *Obsérvese que* $t(\alpha)$ *é necesariamente a n-forma normal (por* $\succ_n$*) de* σ *e tamén se dirá a n-forma minimal de* σ,
- *non está definido noutro caso.*

O n-grado *dun par de substitucións* σ, τ, *é a suma dos seus n-grados. O* n-grado *de* θ *respecto de* σ, τ, *expresado por n-grado*$(\sigma \overset{?}{=} \tau, \theta)$ *é :*

- *0 se* $\sigma\theta = \tau\theta$,
- m= *n-grado*$(\sigma\theta, \tau\theta)$.

Para evitar o problema da non completitude de narrowing condicional respecto das solucións normalizadas (exemplo *4.11*), emprégase a n – *minimalidade*, para a cal sí que se ten a completitude, gracias a unha versión do teorema de Hullot. Nótese que, naquel exemplo, a solución $\{x/f(a), y/a\}$ é 0-minimal.

4.2 *Sexan $n \in N$, $I \overset{\sigma,\tau}{\rightarrow}$ e R un conxunto de 2-células condicionais, pechado e dirixido por $\succ$, sendo ésta unha relación completa por niveles. Sexa $X \xrightarrow{\theta} Y$ unha substitución n-minimal tal que existe unha 2-célula de profundidade n, que iguala $\sigma\theta$ e $\tau\theta$. Entón, existe $\theta' \in S(\sigma \overset{?}{=} \tau,\ \overset{nc}{\Rightarrow}),\quad \theta' \leq \theta$.*

Proba. Inducción en n. Para $n = 0$, $\theta \in Unif(\sigma \overset{?}{=} \tau)$ e $\theta' = mgu(\sigma, \tau) \in S(\sigma \overset{?}{=} \tau,\ \overset{nc}{\Rightarrow})$. Suposto para $n-1$. Sexa $m = n - grado(\sigma \overset{?}{=} \tau, \theta)$. Pola confluencia por niveles de $\succ$, pode supoñerse que existen 2-células $\begin{array}{ccc} \sigma\theta & & \\ \downarrow\bar{\sigma} & ex(\alpha) & \\ & \searrow & \end{array}$ $\begin{array}{ccc} \bar{\sigma} & & \\ \downarrow\phi & * & \\ & \searrow & \end{array}$, $\begin{array}{ccc} \tau\theta & & \\ \downarrow\phi & * & \\ & \searrow & \end{array}$, obtidas mediante unha estratexia *innermost*.

Se probará que existe un paso $\sigma \overset{nc}{\Rightarrow}_\rho \sigma'$ e unha substitución λ'' n-minimal, tal que

i) $\sigma'\lambda'' = \bar{\sigma},\ \rho\lambda'' = \theta$,

ii) $n - grado(\sigma' \overset{?}{=} \tau\rho, \lambda'') < m$.

Xa que θ é n-minimal, a 2-célula α dada por $\left(I \overset{s(\alpha)}{\underset{t(\alpha)}{\Rightarrow}} X_\alpha,\quad I' \overset{c_\alpha,c'_\alpha}{\rightarrow} X_\alpha + X^e_\alpha\right)$ é aplicada nunha posición π de σ e a substitución correspondente a esta posición, non é isomorfismo na imaxe. Polo tanto, existe unha substitución $\psi : X_\alpha + X^e_\alpha \rightarrow Y$ tal que:

$$\sigma\theta/\pi = s(\alpha) \circ j_{X_\alpha} \circ \psi$$

$$c_\alpha \circ \psi =_R c'_\alpha \circ \psi$$

A substitución ψ pode supoñerse que é *n-minimal* : para X_α pola selección da estratexia *innermost* e para as variables de X^e_α pola confluencia por niveles.

Da primeira ecuación dedúcese a existencia de $\delta : X + X_\alpha \rightarrow Z,\ \delta = mgu(\sigma/\pi \circ j_X, s(\alpha) \circ j_{X_\alpha})$ e dunha substitución $\lambda : Z + X^e_\alpha \rightarrow Y$, tal que $<\delta \circ j_Z, j_{X^e_\alpha}> \lambda = <\theta, \psi>$. Pola n-minimalidade de θ e ψ, a substitución λ verifica as hipóteses establecidas neste caso para $n-1$ respecto do problema

$$c_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ\ <\delta \circ j_Z, j_{X^e_\alpha}> \overset{?}{=} c'_\alpha \circ j_{X_\alpha + X^e_\alpha} \circ\ <\delta \circ j_Z, j_{X^e_\alpha}> .$$

Dado que se verifica el resultado para $n-1$, pode supoñerse a existencia de $\lambda', \lambda'',\ \lambda'\lambda'' = \lambda$, onde λ' foi obtido por $\overset{nc}{\Rightarrow}$. Denotando por $\rho = j_X \circ \delta \circ j_Z \circ \lambda'$, dedúcese a existencia dun paso $\sigma \overset{nc}{\Rightarrow}_\rho \sigma'$.

Para probar que se verifica a condición *i)*, obsérvase que :

para as posicións $\omega \perp \pi$, $\quad (\sigma'/\omega)\lambda'' = (\sigma/\omega) \circ j_X \circ \delta \circ j_Z \circ \lambda'\lambda'' = (\sigma/\omega)\theta = \bar{\sigma}/\omega$,

para a posición π : $(\sigma'/\pi)\lambda'' = t(\alpha) \circ j_X X_\alpha \circ \delta \circ j_Z \circ \lambda'\lambda'' = t(\alpha) j_X X_\alpha \psi = \bar{\sigma}/\pi$.

Aplicando o mesmo razonamento que para λ, λ'' é n-minimal, o problema $\sigma' \overset{?}{=} \tau\rho$ é resolto por λ'' e, de toda derivación de $\sigma'\lambda'' =^*_R \tau\rho\lambda''$, obtense unha derivación para

$\bar{\sigma} \stackrel{?}{=} \tau\theta$ e recíprocamente (polas condicións de θ e ψ), polo que o $n\text{-}grado(\sigma' \stackrel{?}{=} \tau\rho, \lambda'') \leq n - grado(\bar{\sigma} \stackrel{?}{=} \tau\theta) < m$.

Para probar finalmente o resultado aplícase inducción no $n - grado(\sigma \stackrel{?}{=} \tau, \theta) = m$. Para $m = 0$ $mgu(\sigma, \tau) \leq \theta$. Suposto para $m - 1$, aplícase *i)*, de onde se obtén $\rho \leq \theta$, e o novo problema $\sigma' \stackrel{?}{=} \tau\rho$ unificable por λ'', que é a súa vez *n-minimal* e $n - grado(\sigma' \stackrel{?}{=} \tau\rho, \lambda'') < m$. Por inducción en m obtense $\rho' \in S(\sigma' \stackrel{?}{=} \tau\rho, \stackrel{nc}{\Rightarrow})$, $\rho' \leq \lambda''$. Polo tanto, $\rho\rho' \leq \theta$, é *n-minimal*, obtido por narrowing condicional □.

4.3 *Sexa R un conxunto de 2-células pechado e dirixido pola relación $\succ$ que é completa por niveles. Narrowing condicional e narrowing condicional demorado proporcionan conxuntos completos de R-unificadores de todo problema de E-unificación dado.*

Proba. Pola completitude por niveles, para todo R-unificador θ de $\sigma \stackrel{?}{=} \tau$ e se n é a profundidade dunha derivación $\overset{\sigma\theta}{\phi \downarrow *}$, $\overset{\tau\theta}{\phi \downarrow *}$, existe unha substitución $\theta' =_R \theta$ que é *n-minimal*. O resultado séguese directamente de aplicar a propiedade anterior.

Para o narrowing condicional demorado, o resultado é certo xa que toda secuencia de narrowing condicional que produce $\theta \in S(\Gamma, \stackrel{nc}{\Rightarrow})$ pode adaptarse a unha secuencia de narrowing condicional demorado. □.

Nótese que as derivacións empregadas na proba anterior, que producen o E-unificador θ son de feito secuencias de *basic narrowing*. Esto é certo xa que as θ' son *n-minimais* e a derivación que iguala $\sigma\theta'$ e $\tau\theta'$ é *innermost*. Non cabe, polo tanto, reducir as instancias das variables das 2-células. Por outra parte, as posicións nas que se ten aplicado narrowing, coinciden coas posicións empregadas na derivación e son polo tanto tamén posicións básicas.

4.3 E-unificación en teorías xerais

4.3.1 Algoritmos xerais de Unificación Ecuacional

Un tratamento completo da E-unificación, pasa por dispor de algoritmos de E-unificación en casos máis xerais que os mencionados. Amais, é de gran interese coñecer as xeralizacións ás que hai que someter os anteriores algoritmos para incorporar outras teorías. O problema da explosión combinatoria dificulta enormemente a súa posible utilización, de aí o aínda maior interese por eliminar as redundancias nestos casos xerais.

En principio, para tratar coas ecuacións no marco da proba automática, compriu incorporar os axiomas de igualdade ós axiomas propios da teoría. Para ese conxunto de axiomas a regra de resolución é un procedemento de E-unificación. A primeira regra específica para o predicado de igualdade é a *paramodulación*[153] e variantes de ésta como a *hiperparamodulación, paramodulación lineal* e outras [38]. Como indican Wos e McCune[186]. :

> *Indeed, although we are very partial to the treatment given to equality by the use of the inference rule paramodulation, we apply recognize that this inference rule does not by itself solve all the problems that accompany a natural treatment of equality. As a result, we are still actively searching for ever more powerful strategies to control the application of paramodulation.*

Lankford [110] probou que a paramodulación non necesita dos axiomas reflexivos funcionais. Os maiores problemas do uso da paramodulación, para obter procedementos efectivos de E-unificación son :

- a necesidade de permitir a paramodulación nun factor. Para elo, se t é un termo dado, a paramodulación pódese realizar en calesquera expresións $t\sigma$ onde σ é unha instanciación arbitraria,
- a dificultade de conservar a completitude se as ecuacións son empregadas en forma de reescrita, esto é só en nun dos dous sensos.

No caso de teorías para as que a igualdade pode sustituirse por unha relación de reescrita coa finitariedade e a confluencia dos termos concretos cunha determinada orientación de os axiomas, Nutt *et al.* [135] proban que *basic narrowing* proporciona aínda conxuntos completos de E-unificadores. En casos máis xerais, parece preciso empregar os axiomas nos dous sensos, e que sexa o propio mecanismo de inferencia o que se encargue de eliminar redundancias e bucles creados polo uso de regras sen orientar.

As condicións baixo as cales faise necesario aplicar a paramodulación nalgún factor, atópanse moi ligadas á a existencia de pares críticos. Sexa t un termo e $l\rho[\pi \leftarrow r'\rho] \rightarrow r\rho$ un par crítico formado a partir de $l \rightarrow r$ e $l' \rightarrow r'$. Entón se se verifican as seguintes condicións :

- existe unha sustitución σ, $t\sigma[\pi \leftarrow r'\rho] = l\rho\sigma$,
- t e l non son unificables,
- π é unha posición variable de t ou $\pi > \pi'$ onde π' é unha posición variable de t,

entón pode aplicarse $\stackrel{\pi}{\Rightarrow}$ en t usando o par crítico e sen embargo nen $l \rightarrow r$ nen $l' \rightarrow r'$ poden ser utilizadas para sustituir esa aplicación. Para evitar tal situación, a paramodulación pode ser utilizada en posicións creadas por instanciación do termo dado ([38]).

Exemplo 4.12 ([71]) $R = \{a \rightarrow b,\ f(g(a),g(b)) \rightarrow h(g(a),g(b))\}$, $\Gamma = \{f(x,x) \stackrel{?}{=} h(x,x)\}$. *Para obter a solución* $\sigma = \{x/g(a)\}$, *debe aplicarse a paramodulación na posición* $1 \cdot 1$ *creada previamente por instanciación (regra* $\rightarrow_{ip(\sigma)}$ *en Hölldobler), xa que* $f(x,x)$ *e* $f(ga,gb)$ *non son unificables. Nótese que nen sequera permitindo aplicar* $a \rightarrow b$ *nunha posición da variable* x *resólvese o problema.*

Gallier e Snyder [71], para obter un procedemento completo en teorías xerais, adaptaron o enfoque de Kirchner[103], pero debilitando a etapa de *Mutación.* Empregan para elo, dúas novas regras de transformación que se incorporan a $\mathcal{ST}$. A primeira regra vén dada por :

Root-Rewriting $\Gamma \cup \{s \stackrel{?}{=} t\} \Rightarrow \Gamma \cup \{s \stackrel{?}{=} l\} \cup \{r \stackrel{?}{=} t\}$ onde :

- $l = r$ é unha variante dunha ecuación de E, $Var(l,r) \cap (Var(\Gamma) \cup Var(s,t)) = \emptyset$,
- s non é variable e se l non é variable, entón *operador superior (s)= operador superior(l)*,
- Esta transformación non pode ser aplicada en nengunha expresión resultante de $s \stackrel{?}{=} l$.

A segunda regra é empregada para eliminar ciclos :

Imitate $\Gamma \cup \{x \stackrel{?}{=} t\} \Rightarrow \Gamma \cup \{x \stackrel{?}{=} f(y_1, \ldots, y_n)\} \cup \{f(y_1, \ldots, y_n) \stackrel{?}{=} t\}$ onde

- $x \in Var(t)$, $t = f(t_1, \ldots, t_n)$,
- $y_1, \ldots, y_n$ son novas variables non existentes en $\Gamma \cup \{x \stackrel{?}{=} t\}$,
- aplícase inmediatamente a *Eliminación de Variables* en $\{x \stackrel{?}{=} f(y_1, \ldots, y_n)\}$ e a *Descomposición* en $\{f(y_1, \ldots, y_n) \stackrel{?}{=} t\}$.

Para evitar a aparición de secuencias infinitas por aplicación de *Imitate*, non se permite unha nova aplicación de ésta antes que *Root Rewriting* sexa aplicada nalgún dos pares procedentes de $f(y_1, \ldots, y_n) \stackrel{?}{=} t$ que corresponden a posicións anteriores ás $\in \Pi_x(t)$.

Os mecanismos de E-unificación que non fan uso de axiomas no interior dun termo antes de que teñan sido eliminados os seus operadores previos, son chamados *top-down* e explotan o feito de que a regra *Descomposición* reduce paulatinamente os operadores superiores. Pódense ver como xeralizacións do método de Kirchner onde a regra de mutación áchase moi debilitada.

Gallier e Snyder [72] sustituen a regra *Imitate* por *Root Imitation* permitindo que t sexa variable. Recentemente Delsart [50] mostra un novo método de E-unificación baseado nunha estratexia *top-down* para teorías que poden ser dadas por presentacións condicionais.

Para elo, substitúe as transformacións *Root Rewriting* e *Imitate* por :

Mutate $\Gamma \cup \{s \stackrel{?}{=} t\} \Rightarrow \Gamma \cup \{r\sigma \stackrel{?}{=} t\sigma\} \cup \{e_1\sigma \stackrel{?}{=} e'_1\sigma, \ldots, e_m\sigma \stackrel{?}{=} e'_m\sigma\} \cup \underline{\sigma}$ onde

- $e_1 \stackrel{?}{=} e'_1, \ldots, e_m \stackrel{?}{=} e'_m \Rightarrow l \rightarrow r$ é unha variante dunha regra ecuacional,
- $\sigma = mgu(s,l)$,

- $\underline{\sigma} = \{x_1 \stackrel{?}{=} s_1 \ldots, x_n \stackrel{?}{=} s_n\}$ é a forma resolta correspondente a $\sigma = \{x_1/s_1, \ldots, x_n/s_n\}$.

Neste caso, para que o método sexa completo, é preciso utilizar *presentacións estrictamente resolventes*[9], polo que o método de Delsart achégase máis aínda ó de Kirchner[103]. Delsart proba que toda teoría ten unha presentación estrictamente resolvente. O principal problema é que, para o caso xeral, esta presentación incrementa o número de axiomas e, para cada un de eles, un dos termos incorpórase á parte condicional (quedando no seu lugar unha variable). Así, convirte a ecuación $l = r$ nas ecuacións condicionais $x \stackrel{?}{=} l \Rightarrow x = r$ e $x \stackrel{?}{=} r \Rightarrow x = l$ onde x é unha nova variable e incorpora a restricción (análoga á establecida por Gallier e Snyder) de que *Mutate* non sexa novamente aplicada en $r\sigma \stackrel{?}{=} t\sigma$. Delsart proba que nun problema do tipo $s \stackrel{?}{=} t$, non é necesario que *Mutate* sexa aplicada en ambos membros, obtendo unha forma de eliminar algunhas das redundancias creadas pola non direccionalidade das ecuacións.

Polo tanto e, agás para certas teorías, as condicións impostas para aplicar reglas no método de Delsart son como as do método de Gallier e Snyder, e incorporando as restriccións da dirección. Entre as teorías para as que Delsart mostra presentacións estrictamente resolventes, atópanse as teorías sintácticas, tamén aquelas que se obteñen combinando unha teoría sintáctica con outra teoría con operadores libres e a teoría de axiomas conmutativo e transitivo[10].

Gallier e Snyder[72] nun traballo posterior, introduciron un novo conxunto de transformacións (aquí expresado por $\mathcal{T}_{\mathcal{CP}}$) que inclúe $\mathcal{ST}$ e unha nova regra

Lazy Paramodulation $\Gamma \cup \{s \stackrel{?}{=} t\} \stackrel{lp}{\Rightarrow} \Gamma \cup \{s/\pi \stackrel{?}{=} l,\ s[\pi \leftarrow r] \stackrel{?}{=} t\}$ onde :

- (l, r) é unha variante dunha regra ecuacional,
- s/π non é variable e l é variable ou ben *operador superior*(s/π) =*operador superior*(l),
- na expresión $s/\pi \stackrel{?}{=} l$ aplícase inmediatamente a transformación *Descomposición* que elimina o operador superior (se l non é variable).

Gallier e Snyder proban a completitude do novo método baixo a seguinte restricción:

> *Lazy Paramodulation* só se emprega na posición <> cando ou ben s ou ben t é variable.

Para probar a completitude de $\mathcal{T}_{\mathcal{CP}}$ baixo esta estratexia *top-down*, asóciase a cada E-unificador unha árbore de proba. Esta árbore ten na súa raíz o par de termos instanciados polo E-unificador; as súas follas corresponden a igualdades sintácticas; as

[9] aquelas para as que se verifica $\stackrel{+}{\rightarrow}_R \subseteq \rightarrow_R \stackrel{*}{\longrightarrow}_R \stackrel{\neq <>}{}$

[10] o axioma transitivo é $(xRy) * (yRz) = (xRy) * (xRz)$.

líneas se constrúen ben pola eliminación dos operadores comúns en ambos termos ou ben por aplicación de axiomas. Sobre esta árbore, actúan un conxunto de transformacións coa propiedade de proporcionar novas árbores terminais que teñen na súa raíz unha forma resolta. As transformacións sobre a árbore permiten construír unha derivación do problema inicial usando as reglas de $\mathcal{T}_{\mathcal{CP}}$. Mediante un esquema semellante Nutt *et al.*[135] proban, en sistemas confluentes concretos, a completitude dunha versión de *basic narrowing* con regras de fallo e simplificadores.

Gallier e Snyder [72] atoparon outra forma de probar a completitude do conxunto de regras $\mathcal{T}_{\mathcal{CP}}$. Sepáranse os axiomas en R, E (ecuacións orientadas e axiomas sen orientación respectivamente) e pártese dun orden de reducción $\gg$ total nos termos concretos[11]. Realízase un proceso de complección de pares críticos (realmente non é necesario un procedemento particular, nen tampouco a súa terminación) que proporciona un conxunto $E^{\omega} \cup (E^{\omega})^{-1} \cup R^{\omega}$. Proban que este conxunto ten a propiedade de ser Church-Rosser concreto relativo a $\gg$ [12].

Con ese conxunto de ecuacións, o procedemento de narrowing usando ecuacións de $E^{\omega} \cup (E^{\omega})^{-1} \cup R^{\omega}$ proporciona un conxunto completo de $(E \cup R)$-unificadores concretos. Por outra parte, toda secuencia de narrowing pode ser incluída como unha secuencia de $\mathcal{T}_{\mathcal{CP}}$. Neste punto, proban (lema 6.7) que para $\mathcal{T}_{\mathcal{CP}}$ a complección por pares críticos non é necesaria (e de aí a xustificación de que a complección non necesite un procedemento dado). Para elo, toda secuencia de regras de $\mathcal{T}_{\mathcal{CP}}$ que proporcione un E-unificador e use pares críticos pode ser substituída por unha secuencia destas mesmas $\mathcal{T}_{\mathcal{CP}}$ empregando axiomas de $E \cup E^{-1} \cup R \cup R^{-1}$. A forma de pasar de E-unificadores concretos a E-unificadores xerais, obtense incluíndo unha cantidade numerable de novas constantes (unha para cada variable) e, como consecuencia da pureza do procedemento (non emprega operadores que non estean presentes no problema ou nas ecuacións), todo $(E \cup R)$-unificador correspóndese cunha substitución concreta. Os métodos que aplican as regras imitando narrowing, e polo tanto a propia derivación de reescrita, son chamados métodos *down-up*. Só se apartan da derivación para reemplazar o uso dos correspondentes pares críticos aplicados.

Esta forma de probar a completitude deste método significa un novo enfoque nos algoritmos de E-unificación. Ohsuga e Sakai[136] empregan igualmente a complección sen fallo para obter un sistema completo de E-unificadores, sendo o seu método unha combinación estándar de pasos de narrowing acompañados da complección por pares críticos e engadindo regras de simplificación. O aplicar a complección paralelamente á E-unificación, nalgúns casos poderán xerarse un número infinito de pares críticos sen trascendencia algunha para o problema de E-unificación dado. En cambio coas transformacións $\mathcal{T}_{\mathcal{CP}}$ xérase unha forma de narrowing suficientemente débil como para sustituir só os pares críticos que poden ser usados na obtención dos E-unificadores.

[11] un orden de reducción é un orden monótono, estable e noetheriano.

[12] E, R é Church-Rosser concreto relativo a $\gg$ se dados termos s, u, t concretos tales que $s \leftarrow_{E,R} u \rightarrow_{E,R} t$, existen instancias concretas de R e E (orientadas no senso de $\gg$) de forma que $s \rightarrow_{R(E),R} v \leftarrow_{R(E),R} t$, onde $R(E) = \{l\sigma = r\sigma | l\sigma \gg r\sigma \in E \cup E^{-1}\}$.

Pode observase que a separación dos axiomas en E, R non ten consecuencias prácticas, ó ser necesario empregar tamén os elementos de R^{-1}. Por outra parte cabe agardar unha mellora do algoritmo usando regras con condicións máis fortes que *Lazy Paramodulation* que aínda permitan imitar o uso por *narrowing* dos pares críticos.

Os intentos realizados para evitar o uso de R^{-1} mostran a dificultade de orientar os axiomas nun caso xeral. Aínda separando os axiomas en regras orientables e ecuacións no lema 6.7 de Gallier e Snyder[72] precísase empregar as regras e ecuacións nos dous sensos. Amais, existen problemas que requiren o uso de ecuacións nas dúas expresións dun problema dado, debido esencialmente á prohibición de aplicar *Lazy Paramodulation* en variables.

Exemplo 4.13 $R = \{f(y) \xrightarrow{\alpha} y,\ f(h(y)) \xrightarrow{\beta} y\},\quad \Gamma = \{h(x) \stackrel{?}{=} x\}.$
A secuencia de Lazy Paramodulation que proporciona a solución $\varepsilon = 1_{\{x\}}$ *é*
$h(x) \stackrel{?}{=} x \Rightarrow_{\alpha^{-1}} f(h(x)) \stackrel{?}{=} x \Rightarrow_{\beta} x \stackrel{?}{=} x,$ *onde se fai necesario empregar* α^{-1} *pese á unicidade das orientación de* α.

A continuación móstrase un problema para o que é preciso aplicar regras nos dous termos :

Exemplo 4.14 $R = \{f(y) \xrightarrow{\alpha} y,\ f(h(y)) \xrightarrow{\beta} y,\ f'(y) \xrightarrow{\alpha'} y,\ f'(h'(y)) \xrightarrow{\beta'} y\},$
$\Gamma = \{h(x) = h'(x)\}$
$h(x) \stackrel{?}{=} h'(x) \Rightarrow^{*}_{\alpha^{-1},\alpha'^{-1}} f(h(x)) \stackrel{?}{=} f'(h'(x)) \Rightarrow^{*}_{\beta,\beta'} x \stackrel{?}{=} x.$

Moser [129] define un novo conxunto de regras de transformación para a unificación ecuacional, empregando as regras *Unification* e *Lazy Basic Paramodulation.*

Unification

$$\frac{\left(R \cup \{s \not\simeq t\}\right) \cdot \sigma}{R \cdot \sigma\theta} \qquad \text{onde } \theta = mgu(s\sigma, t\sigma)$$

Lazy Basic Paramodulation

$$\frac{\left(R \cup \{s \not\simeq t\}\right) \cdot \sigma}{\left(R \cup \{s/\pi \not\simeq l, s[\pi \leftarrow r]\}\right) \cdot \sigma} \quad \begin{array}{l}\text{onde } \pi \in \Pi_{\Sigma}(s) \\ (l \simeq r) \in E \text{ e se } l \notin V \ \textit{operador-superior}(s/\pi) = \\ \textit{operador-superior (l)}\end{array}$$

A primeira extrae directamente o unificador máis xeral dun problema simple (par de termos a unificar) e compónno coa instanciación previa, pero sen instanciar o resto do problema. Polo tanto emprega en forma non determinística a regla que extrae un unificador sintáctico e, a diferencia de $\mathcal{T}_{\mathcal{LP}}$, non emprega expresamente a *Descomposición.* A segunda é semellante á regra *Lazy Paramodulation* aplicada a este contexto de pares *(problema actual,instanciación acumulada).* O principal resultado de Moser é que se conserva a completitude empregando conxuntamente as seguintes condicións :

- non instanciación do problema actual, gardando separadamente as instanciacións recollidas,
- as regras de conmutación poden ser aplicadas sen perder a completitude. Amais dase a conmutación forte (empregar dos transformacións seguidas en posicións independentes é conmutable),
- a obtención directa do *mgu* sen precisar do uso das súas regras en forma separada.

A diferencia de *Lazy Paramodulation*, no par $s/\pi \not\preceq l$ non se pode aplicar inmediatamente a transformación *Descomposición.*

A proba da completitude obtense a partir dunha variante do cálculo denominado *basic superposition*[14]. Este cálculo emprega a regra *Positive Constrained Basic Paramodulation* que é equivalente a unha forma de xerar pares críticos. Proba que os E-unificadores obtidos por este cálculo poden ser reemplazados polo uso das transformacións *Unification* e *Lazy Basic Paramodulation.* Polo tanto, o método de E-unificación de Moser é do tipo *down-up.* Comparado con $\mathcal{T}_{\mathcal{LP}}$ é máis efectivo fundamentalmente gracias tanto ó emprego da *conmutación* como por gardar as instanciacións producidas, se ben non incorpora a restricción de *Lazy Paramodulation* da eliminación do operador superior en $s/\pi \stackrel{?}{=} l$. Por outra parte, unha estratexia de demora das instanciacións pode repercutir na detección tardía da non unificabilidade.

Exemplo 4.15 R=$\{f(a) \rightarrow h(x,x),\ h(b,b) \rightarrow g(b,b)\}$, $\Gamma = \{f(x) \stackrel{?}{=} g(x,x)\}$. *A detección da non unificabilidade de* Γ *é máis lenta por un procedemento de demora que por unha secuencia que aplique a instanciación* $\{x/a\}$ *ó problema.*

Dougherty e Johann[56] reemplazan a regra *Lazy Paramodulation*, por unha nova regra *Relaxed Paramodulation* moito máis restrictiva. O novo conxunto de transformacións, aínda permite reemplazar secuencias de narrowing con ecuacións procedentes do procedemento de complección por pares críticos, por secuencias de regras deste conxunto usando ecuacións dunha presentación arbitraria.

Definición 4.9 *Sexan s e t termos. Dise que s e t satisfán a condición* top-unify *se se verifica unha das seguintes condicións :*

- *o termo s ou o termo t é variable,*
- $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ *e para todo* $i, 1 \leq i \leq n$, *top-unify*(s_i, t_i)

A nova regra é a seguinte :

Relaxed Paramodulation $\Gamma \cup \{s \stackrel{?}{=} t\} \stackrel{rp}{\Rightarrow} \Gamma \cup \{s/\pi \stackrel{?}{=} l, s[\pi \leftarrow r] \stackrel{?}{=} t\}$ se :

- (l,r) é unha variante dunha ecuación de E,
- s/π non é variable e verifícase *top-unify*$(s/\pi, l)$

O método de Dougherty e Johan é chamado *Relaxed Narrowing* e emprega as regras de $\mathcal{ST}$, xunto coa regra *Relaxed Paramodulation* baixo a seguinte restricción :

> Logo de aplicar a transformación *Relaxed Paramodulation*, aplícase a regra *Descomposición* tantas veces como sexa posible, ata eliminar os operadores comúns nambos membros.

Como o propio nome indica, é un algoritmo do tipo *down-up* cunha maior aproximación ó procedemento de narrowing.

Pode verse na regra *Relaxed Paramodulation* unha profundización da condición de *Lazy Paramodulation*, ó realizar unha comprobación cara adentro dos operadors ata asegurá-la non existencia de conflictos (*clash*). Esto contrasta co único test se realiza en *Lazy Paramodulation* exclusivamente para os operadores superiores de s/π e l. O método de Dougherty e Johann emprega un control máis estricto nos operadores realizando as derivacións máis próximas ó narrowing.

Xa que *Relaxed Paramodulation* é moito máis restrictiva que *Lazy Paramodulation* é un problema de interese obter novas restriccións que conserven a completitude eliminando derivaciones derivacións innecesarias, ben porque non teñen solución ou ben porque éstas xa se atopan producidas por outras derivacións. O método que se expón a continuación incorpora algunhas das modificacións de Moser, pero sobre todo impón condicións máis restrictivas que *Relaxed Paramodulation.* Por outra parte, próbase que unha adaptación de *Relaxed Narrowing* permite amais o emprego de presentacións condicionais.

4.3.2 E-Unificación de substitucións en teorías xerais

Abstracción de termos e substitucións

O resultado máis relevante do presente traballo consiste na proba de que o procedemento de Dougherty e Johann pode ser sustancialmente mellorado mediante a imposición de condicións máis fortes. Amais pódense incorporar ecuacións condicionais e engadir algunhas novas restriccións sen perder a completitude. Para elo, se verá que o procedemento coñecido como *abstracción* permite realizar a E-unificación de pares de substitucións en teorías xerais, extendendo os mencionados resultados de Rydeheard e Stell en teorías canónicas.

A abstracción de termos ten sido empregada para debilitar o problema inicial e, a partir das novas solucións (más xerais que as do problema inicial), confrontar a instanciación creada coa parte demorada ata obter as solucións requeridas.

A abstracción aparece definida en Stickel[176] e é utilizada para conseguir a homoxeneidade dos termos[13]. Yelick[188] emprega a abstracción na unificación en teorías combinadas, para separar o problema en subproblemas que poden ser resoltos

[13]un termo nunha signatura con operadors AC, é homoxéneo se o *operador superior* é AC e non ten subtermos encabezados con outro operador.

separadamente nas subteorías, procedendo a continuación a reunir as instanciacións cos problemas demorados. Na Unificación AC e na E-unificación de teorías combinadas téñense utilizado tamén outras formas de abstracción : a de Lincoln e Christian[114] que permite separar as variables repetidas; Schmidt-Schauß [164] emprega tanto unha forma débil de abstracción (conserva as constantes libres) como unha forma forte na que reemplaza tamén constantes libres.

Tamén en Alpuente *et al.* [4] aparece unha forma algo especial de abstracción. Neste caso todo subtermo do lado dereito ou condicional dunha regra que poida ser problemático, no senso de que a aplicación de narrowing nil pode xerar a non terminación de narrowing[14], é reemplazado por un símbolo terminal $\perp$. Este símbolo terminal ten a finalidade de eliminar tales subtermos que poden impedir a terminación do algoritmo.

Noutros casos, a abstracción é menos explícita, como no traballo de Nipkow [133], na unificación de Alxebras Primais[15]; Giovannetti *et al.* [74] empregan unha forma de abstracción para pasar dun procedemento basado en narrowing a un procedemento por SLD-resolución; Baader realiza a abstracción de constantes (reemplazadas por novas variables) para obter os E-unificadores nunha teoría conmutativa unitaria se se incorporan constantes libres; Kirchner[103] emprega a *xeralización*, para obter termos sen operadores nos argumentos, reemplazando momentáneamente estes por novas variables. A seguinte tabla, móstranos algúns tipos de abstracción empregados en distintos contextos :

Autor– Contexto	*Expresión previa*	*Expresión nova*	*Condición demorada*
Stickel $Unif.As - Comm$	$t = f(..,s,..,s,..)$ $s = g(\ldots), f \in \Sigma_{AC}$ s maximal, $g \neq f$	$\bar{t} = f(..,x',..,x',..)$	$x' = s$
Yelick *Combinación de Teorias*	$t = f(.,s,.,s,.)$ $f \in \Sigma_i, s = g(..)$ s maximal, con $g \notin \Sigma_i$	$\bar{t} = f(.,x^1,.,x^n,.)$	$x^1 \stackrel{?}{=} s$ $\ldots$ $x^n \stackrel{?}{=} s$
Lincoln– Christian $Unif.As - Comm$	$t = f(.,s,.,s,.)$ $f \in \Sigma_{AC}$ e $s = g(.)$ ó $s = x$	$\bar{t} = f(.,x^1,.,x^n,.)$	$x^1 \stackrel{?}{=} s$ $\ldots$ $x^n \stackrel{?}{=} s$
Alpuente et al. *Chequeo Inconsistencia*	$l \to r = g(.,s,.)$ $s = f(.)$ maximal en r $f \in$ ciclo en G_R	$l \to \bar{r} = g(.,\perp,.)$	*modifica condicción mgu*

Taboa 4.1. Diversas formas de abstracción en Unificación e Narrowing

[14]esto é detectado mediante un grafo que se crea entre os operadores. A non terminación implica a existencia dun ciclo na compoñente do grafo que contén o operador superior do subtermo.

[15]aquelas para as que toda función finitaria de aridade > 0 exprésase como un termo.

Aquí empregarase unha forma da abstracción que difire algo da utilizada por Lincoln e Christian xa que non se reemplazan operadores :

Definición 4.10 *Chamarase* abstracción *de $\sigma : I \rightarrow X$ ó reemplazamento de toda posición variable de $I\sigma$ por novas variables (unha por posición). A sustitución que leva a toda variable nova a variable á que sustitue, será denominada* substitución da abstracción *e ó reemplazamento o* resultado da abstracción, *esto é :*

Sexan $x_1, \ldots, x_m \in X$ e $\pi_i^1 \ldots, \pi_i^{n_i}$ as posicións en σ tal que $\sigma/\pi_i^j = x_i$. Sexa x_i^j a nova variable que corresponde á posición π_i^j e $X' = \bigcup_{1 \leq i \leq m}(\bigcup_{1 \leq j \leq n_i}\{x_i^j\})$,

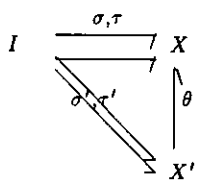
- *a substitución da abstracción $\theta : X' \rightarrow X$ vén dada por $x_i^j\theta = x_i$. Para mostrar a súa relación con σ, θ exprésase tamén por* $\mathrm{abst}(\sigma)$. *A abstracción das substitucións σ e τ indicarase por $abst(\sigma, \tau)$,*

- *o resultado da abstracción $\sigma' : I \rightarrow X'$ é :*
  $\sigma' = \sigma[\pi_1^1 \leftarrow x_1^1] \ldots [\pi_1^{n_1} \leftarrow x_1^{n_1}] \ldots [\pi_i^1 \leftarrow x_i^1] \ldots [\pi_m^{n_m} \leftarrow x_m^{n_m}]$.

Naturalmente $\sigma = \sigma'\theta$. A substitución σ' é a parte debilitada *do problema, mentras que θ é a súa parte* demorada.

Posto que a nova expresión σ' non ten variables repetidas, pode falarse dunha *linearización* das substitucións. A continuación móstrase a solidez e corrección desta operación na unificación ecuacional.

4.4 *Sexan σ, σ', θ, τ, τ' verificando $\sigma = \sigma'\theta$, $\tau = \tau'\theta$.*



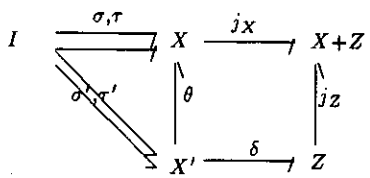
Se $\delta : X' \rightarrow Z$ é un E-unificador de σ', τ'. Entón :

1) se γ é un E-unificador de $\delta \circ j_Z \stackrel{?}{=} \theta \circ j_X$ entón $j_X \circ \gamma$ é un E-unificador de $\sigma \stackrel{?}{=} \tau$,

2) o conxunto $S = j_X \circ \{cU_E(\theta\, j_X, \delta\, j_Z) | \delta \in cU_E(\sigma', \tau')\}$ é un conxunto completo de E-unificadores de $\sigma \stackrel{?}{=} \tau$ (onde $cU(\sigma', \tau')$ é un conxunto completo de E-unificadores de $\sigma' \stackrel{?}{=} \tau'$).

Se amais δ e γ son coigualadores dos repectivos problemas (e.g. se $E = \emptyset$), entón :

3). $j_X\, \gamma$ é un coigualador de σ, γ



1). Dado γ tense que $\sigma \circ j_X \circ \gamma = \sigma'\theta \circ j_X \circ \gamma =_E \sigma'\delta \circ j_Z \circ \gamma =_E \tau'\delta \circ j_Z \circ \gamma =_E \tau'\theta \circ j_X \circ \gamma = \tau \circ j_X \circ \gamma$.

2). A corrección de S queda probada co anterior resultado. Para a completitude de S, sexa $X \xrightarrow{\lambda} Y, \lambda \in Unif_E(\sigma,\tau)$. Entón $\theta\lambda \in Unif_E(\sigma',\tau')$ polo que existen $X' \xrightarrow{\delta} Z, Z \xrightarrow{\rho} Y$, $\delta \in cU_E(\sigma',\tau')$, $\delta\rho = \theta\lambda$. Polo tanto, $< \lambda,\rho > \in Unif_E(\theta \circ j_X, \delta \circ j_Z)$ xa que $\theta \circ j_X < \lambda,\rho >= \theta\lambda = \delta\rho = \delta \circ j_Z < \lambda,\rho >$. De onde se deduce a existencia de $\gamma \in cU_E(\theta \circ j_X, \delta \circ j_Z), \gamma \leq_E < \lambda,\rho >$, e polo tanto $j_X\gamma \leq_E \lambda$.

3). Sexa λ un E-unificador de σ,τ. Xa que $\theta\lambda$ é un E-unificador de σ',τ', existe un único ρ tal que $\delta\rho = \theta\lambda$. Polo tanto, $\theta \circ j_X, \delta \circ j_Z$ son unificados por $< \lambda,\rho >$. Dado que, por hipótese, γ é coigualador, existe un único ρ' tal que $\gamma\rho' =< \lambda,\rho >$ e polo tanto, $j_X \circ \gamma\rho' = \lambda$. Sexan ρ_1,ρ_2 tales que $j_X \circ \gamma\rho_1 = \lambda = j_X \circ \gamma\rho_2$. Por ser γ unificador de $\theta \circ j_X, \delta \circ j_Z$ entón $\theta \circ j_X\gamma\rho_1 = \delta \circ j_Z\gamma\rho_1 = \theta\lambda = \delta\rho$. Pola unicidade de ρ, $j_Z \circ \gamma\rho_1$ é única. De onde se deduce que $\gamma\rho_1$ é xustamente $< \lambda,\rho >$. Pero xa se observou que só existe un $\rho' = \rho_1 = \rho_2$ que verifica $\gamma\rho' = < \lambda,\rho >$ □.

Unha versión deste resultado é empregado por Baader[8] para obter E-unificadores en presencia de constantes libres cando a teoría é conmutativa e de E-unificación unitaria sen estas constantes.

A seguinte propiedade reduce o problema da E-unificación á solución de problemas do tipo $\Gamma = \{i\} \xrightarrow{\sigma,\tau} Y$.

4.5 . *Sexa* $\Gamma = \{I' + I'' \xrightarrow{\sigma,\tau} X\}, \theta : X \rightarrow Y$, $\Gamma' = j_{I'} \circ \Gamma, \Gamma'' = j_{I''} \circ \Gamma$. *Entón* $\theta \in Unif_E\Gamma$ *sse existen* θ',θ'', $\theta' \in Unif_E\Gamma'$ $\theta'' \in Unif_E(\Gamma''\theta')$ *onde* $\theta = \theta'\theta''$.

Esta é unha forma débil do teorema 3.8 de Rydeheard y Burstall. A necesidade séguese de seleccionar : $\theta' = \theta$ e $\theta'' = id_Y$. Para a suficiencia, sexa $\Gamma = \{\sigma,\tau\}$, $\Gamma' = \{\sigma',\tau'\}$, $\Gamma'' = \{\sigma'',\tau''\}$, $\sigma'\theta' =_E \tau'\theta'$ de onde $\sigma'\theta'\theta'' =_E \tau'\theta'\theta''$ e tamén $\sigma''\theta'\theta'' =_E \tau''\theta'\theta''$. Polo tanto $\theta'\theta''$ unifica σ e τ pola universalidade do coproducto □ .

4.3.3 Abstracción e Unificación Ecuacional

Os procedementos de E-unificación *down-up* que imitan o proceso de narrowing, deben permitir o emprego de regras ecuacionais nunha forma máis xeral que a do propio narrowing, para reemplazar o posible uso de regras obtidas nun proceso de complección por pares críticos. Como se ten visto, no primeiro algoritmo deste tipo só se esixe a igualdade dos operadores superiores do termo onde se aplica e do lado esquerdo

da regra, mentras que no de Dougherty e Johann deben coincidir tódolos pares de operadores que se atopan nas mesmas posicións do subtermo dado e do lado esquerdo da regra. A continuación se verá que a condición pode ser notablemente endurecida, ata esixir a existencia dun unificador sintáctico de a abstracción do subtermo dado e do lado esquerdo da regra empregada, condición bastante máis forte que a que se precisa para que *Relaxed Paramodulation* poida ser aplicada.

Se ben o novo algoritmo pode interpretarse nun esquema habitual de multiconxuntos de pares de termos, tense realizado para pares de substitucións, xa que a finalidade do traballo proposto era a extensión dos resultados de Rydeheard e Stell ó caso de teorías xerais, conservando este obxectivo. Por outra parte e, aída que a notación por veces é lixeramente máis complexa, o uso de substitucións obriga a explicitar os diferentes conxuntos de variables empregados, o que axuda a clarificar o diferente papel das variables do problema dado e das ecuacións, o que está directamente relacionado coa mellora introducida. Este control das variables debe ser reforzado ó incorporar ecuacións condicionais.

Relaxed Narrowing e Unificación de Substitucións

A continuación mostrarase como realizar a unificación de substitucións incorporando o procedemento de *Relaxed Narrowing* a este contexto. Para elo se empregará a abstracción de variables, que na forma definida previamente. Deste xeito evítase o uso da *Descomposición* para eliminar os operadores despois de utilizar *Lazy Paramodulation*, xa que esta operación atópase incluída no unificador do novo problema.

Non se mostra explícitamente a forma de integrar *Relaxed Narrowing* con presentacións condicionais, xa que esto mesmo se realizará expresamente para o novo método. De todas formas, pode notarse, que elo corresponde xa a unha nova mellora respecto de *Relaxed Narrowing.*

Sexa $\Gamma = I \stackrel{\mathcal{E}}{\perp\!\!\!\sim} X$ un problema de E-unificación. Darase unha nova regra $\stackrel{rp}{\Rightarrow}$, que se aplica reiteradamente ós sucesivos problemas ata chegar a un par de eles unificables sintácticamente.

Dase, en primeiro lugar, a transformación que extrae o unificador máis xeral. Se $|I| > 1$, pode seleccionarse $I' \subset I$, obter $\Gamma_{I'} = j_{I'} \circ \Gamma$ e, se $\delta = mgu(\Gamma_{I'})$ obtense un novo problema ou instanciar $j_{I \setminus I'}\Gamma$ por δ (evidentemente poden aparecer outras solucións correspondentes á aplicación de axiomas).

Por elo, en forma algo semellante a [129], pode considerarse a existencia dunha regra da forma :

$$I' + I'' \stackrel{\mathcal{E}}{\perp\!\!\!\sim} X \quad \stackrel{mgu}{\Rightarrow}_{I',\delta} \quad I'' \frac{j_{II'} \circ \sigma \delta}{j_{II'} \circ \tau \delta} Z$$

onde $\delta : X \rightharpoonup Z$ é o unificador máis xeral de $j_{I'} \circ \sigma \stackrel{?}{=} j_{I'} \circ \tau$. Esta operación será expresada igualmente por $\stackrel{mgu}{\Rightarrow}$ se non é relevante indicar o subproblema e o unificador.

Para aplicar no problema de E-unificación $I \stackrel{\mathcal{E}}{\perp\!\!\!\sim} X$ a 2-célula α (ou dito máis

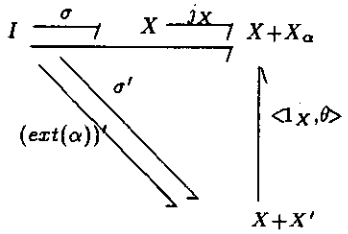
propiamente a 2-célula $ex(\alpha)$ extensión de α), empregarase unha nova transformación $\overset{rp}{\Rightarrow}$. Para observar a forma de actuación de $\overset{rp}{\Rightarrow}$ dividirase o proceso en tres fases :

i) a abstracción dunha parte do problema,

ii) a aplicación dunha forma de narrowing na parte previamente debilitada pola abstracción,

iii) a aplicación da 2-célula e incorporación das condicións demoradas.

i) Sexa σ a substitución do problema onde será aplicada a 2-célula α. En primeiro lugar, realízanse as abstraccións de σ e $s(\alpha)$. Previamente, para conservar a compatibilidade nas variables, é preciso realizar o coproducto $X + X_\alpha$ xa que o novo problema contén ás variables previas (X) xunto coas variables ($X_\alpha = cod(s(\alpha))$. Pode supoñerse que $ex(\alpha)/\pi = \alpha$, polo que a abstracción de σ só se realiza para as variables que se atopan en posicións $> \pi$. Denotando por $\sigma_\pi = \sigma/\pi$, e por σ'_π o resultado da abstracción, a substitución σ será reemplazada por : $\sigma' = \sigma \circ j_X[\pi \leftarrow \sigma'_\pi \circ j_{X'}] : I \rightharpoonup X + X'$, $X' = cod(abst(\sigma_\pi, s(\alpha))$.



Xa que a 2-célula $ex(\alpha)$ coincide con σ nas posicións $\pi', \pi' \perp \pi$ se $X_\alpha = cod(s(\alpha))$, o codominio de $s(ex(\alpha))$ é $X + X_\alpha$, posto que as variables de X poden aparecer nas posicións independentes de π.

ii) A segunda operación consiste na obtención do unificador máis xeral μ do problema $\sigma' \stackrel{?}{=} s(ex(\alpha)) : I \rightharpoonup X + X'$. Dado que as variables de X non son modificadas, μ pode expresarse tamén como $< \delta, 1_X >$, onde $\delta = mgu(\sigma'_\pi, (s(\alpha))') : X' \rightharpoonup Z$, $\mu : X + X' \rightarrow Z + X$.

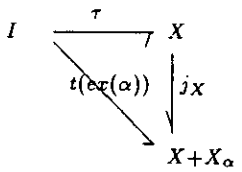
Como consecuencia do teorema de unificación de Robinson[150], se $\gamma = mgu(s,t)$, entón $I(\gamma) \subseteq Var(s,t)$. Análogamente, existe unha inxección $Z \overset{jz}{\rightharpoonup} Z + Z' = X'$. Xeralmente, esta inclusión non aparece explícitamente dado que as variables son renomeadas para garantir a idempotencia.

iii) O último paso consiste na obtención do novo problema que contén como subproblemas a :

a) o par de substitucións obtido ó reemplazar σ por $t(ex(\alpha))$,

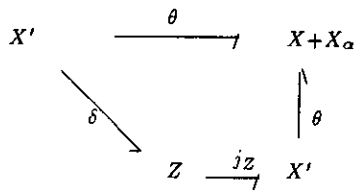
b) as condicións demoradas pola abstracción (a substitución da abstracción, xunto co *mgu* obtido).

As novas substitucións que corresponden a a) son polo tanto $\tau \circ j_X, t(ex(\alpha))$

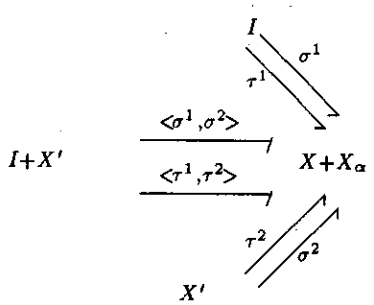


Xa que para toda posición π', $\pi' \perp \pi$, cúmplese que $\sigma/\pi' = t(ex(\alpha))/\pi'$, esta primeira condición do novo problema pode expresarse como $\sigma^1 \stackrel{?}{=} \tau^1$, onde $\sigma^1 = \sigma \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}]$, $\tau^1 = \tau \circ j_X$.

b) A segunda condición, contén a expresión necesariamente demorada dado que non pode ser igualada sintácticamente (a diferencia do narrowing). Esta forma demorada é : $\sigma^2 \stackrel{?}{=} \tau^2$, onde $\tau^2 = \theta : X' \rightarrow X + X_\alpha$ y $\sigma^2 = \delta \circ j_Z \circ \theta$.



Por último, a reunión do par de subproblemas, produce como novo problema :



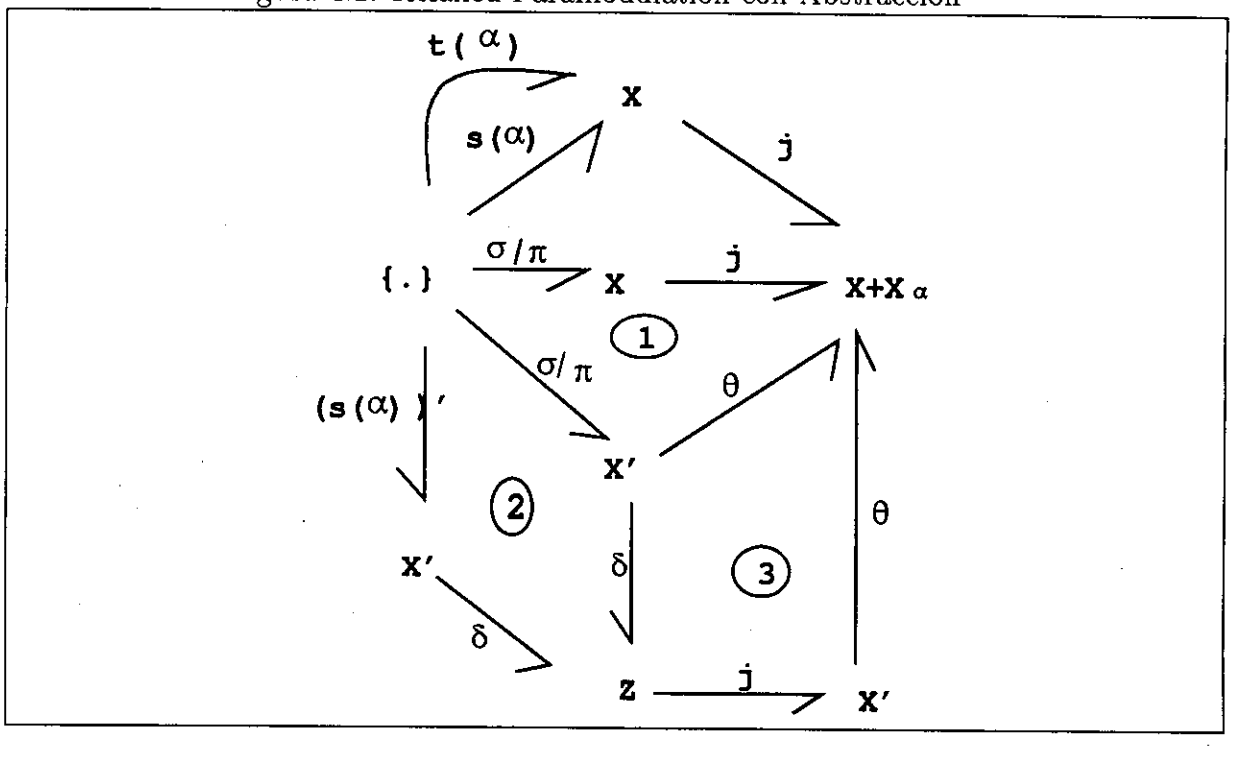
Neste caso, se dirá que $\sigma \stackrel{?}{=} \tau$ é levado mediante *Relaxed Paramodulation con Abstracción* en $<\sigma^1, \sigma^2> \stackrel{?}{=} <\tau^1, \tau^2>$, o que será expresado por

$$\sigma \stackrel{?}{=} \tau \quad \stackrel{ra}{\Rightarrow}_{\pi,\alpha} <\sigma^1, \sigma^2> \stackrel{?}{=} <\tau^1, \tau^2>$$

Obsérvese na figura *4.1* que :

- o diagrama representado por ① corresponde á abstracción
- o correspondente a ② é a extracción do unificador máis xeneral e

Figura 4.1: Relaxed Paramodulation con Abstracción



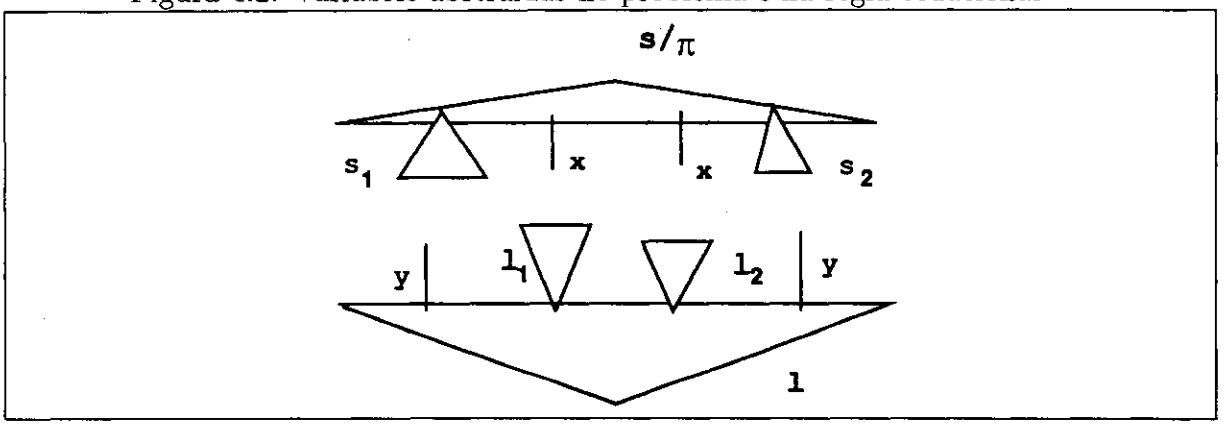
- a operación empregada en ③ trátase da inclusión das variables do codominio do unificador máis xeral e a súa composición coa substitución da abstracción.

No seguinte exemplo móstrase cómo a transformación $\stackrel{rp}{\Rightarrow}$ actúa en forma semellante á propia transformación *Relaxed Paramodulation* logo da eliminación dos operadores imposta por *Relaxed Narrowing.*

Exemplo 4.16 $R = \{f(a,b,x,x) \stackrel{\alpha}{\rightarrow} g(a,b),\ a \stackrel{\beta}{\rightarrow} b\}$, $\Gamma = \{f(y,y,a,b) \stackrel{?}{=} g(a,y)\}$. *Aplicar Relaxed Paramodulation con* α, *produce* $f(y,y,a,b) \stackrel{?}{=} f(a,b,x,x),\ g(a,b) \stackrel{?}{=} g(a,y)$. *Pola restricción imposta, este último problema transfórmase en* $y \stackrel{?}{=} a,\ y \stackrel{?}{=} b,\ x \stackrel{?}{=} a,\ x \stackrel{?}{=} b,\ g(a,b) \stackrel{?}{=} g(a,y)$. *A actuación de* $\stackrel{rp}{\Rightarrow}$ *produce :*

- *etapa de abstracción : o resultado* $f(y_1,y_2,a,b),\ f(a,b,x_1,x_2)$, *e a substitución da abstracción :* $\{y_1/y, y_2/y, x_1/x, x_2/x\}$,
- *extracción do mgu :* $y_1/a,\ y_2/b,\ x_1/a,\ x_2/b$,
- *o novo problema é :* $\{g(a,b) \stackrel{?}{=} g(a,y)\}$ *(correspondente ó reemplazamento)* $\cup\{y \stackrel{?}{=} a,\ y \stackrel{?}{=} b,\ x \stackrel{?}{=} a,\ x \stackrel{?}{=} b\}$ *(consecuencia de compoñer o unificador máis xeral coa substitución da abstracción).*

Figura 4.2: Variables abstraídas no problema e na regla ecuacional



Non se mostra explícitamente a forma e completitude da incorporación de ecuacións condicionais, dado que esto se fará para o novo método de E-unificación. Enfócase agora o problema de obter un método que amais das melloras mencionadas, impoña condicións sustancialmente máis fortes que as exigidas na regra anterior.

4.3.4 Narrowing con Abstracción

Xa que o anterior procedemento actúa sobre as variables X do problema previo e da 2-célula ecuacional X_α mediante a abstracción das variables e posterior extracción do mgu de σ'_π e de $(s(\alpha))'$, prantéxase a cuestión acerca da necesidade desta operación (simétrica respecto de σ e $s(\alpha)$).

Na figura *4.2*, móstrase, coa notación usual, dous termos s/π e l, correspondentes ó subproblema onde se aplica a igualdade $l =_E r$ e a situación das variables repetidas de ambas expresións. Nesta figura *4.2* obsérvase que s_1, s_2 se atopan relacionados cunha mesma variable $y \in Var(l)$, e análogamente l_1, l_2 coa variable $x \in Var(s)$. Supóñase que s_1, s_2 non son unificables, nen tampouco l_1, l_2, así como a existencia dunha regla $l' \to r'$ que aplicada en l_1 permite obter un E-unificador de $l_1 \stackrel{?}{=} l_2$. Esto significa que existe un par crítico formado entre $l \to r$ e $l' \to r'$. Dado que o correspondente par crítico non pode ser explícitamente utilizado, que $l' \to r'$ tampouco pode selo no problema e que l_1, l_2 non son unificables, tense que $l \to r$ só pode ser aplicada en s/π se se realiza a abstracción das posicións variables. Unha vez que l_1, l_2 se atopan como expresións do problema, $l' \to r'$ xa pode ser empregada, permitindo a obtención do correspondente E-unificador.

A abstracción-linearización das posicións da variable $y \in Var(l)$ é necesaria se s_1, s_2 non son unificables sintácticamente. Se tampouco son E-unificables, entón é innecesario permitir a aplicación de $l \to r$. Noutro caso (son E-unificables), existe unha serie de ecuacións que permiten transformar esas expresións ata facer que s_1, s_2 se convirtan en expresións E-unificables. Estas poden aplicarse previamente e, unha vez que as novas expresións s_1, s_2 son E-unificables, xa se pode aplicar $l \to r$. Esto

permite conxeturar que se pode esixir a unificabilidade da expresión previa logo de abstraer as súas variables sen realizar a abstracción das variables das ecuacións. A continuación se verá :

- como realizar esta transformación readaptando o procedemento previamente indicado,
- que o novo sistema permite solucións redundantes producidas por *Relaxed Narrowing*,
- que conserva as propiedades de correción e completitude,
- que permite incorporar ecuacións condicionais,
- que se poden incorporar novas melloras para eliminar algunhas redundancias.

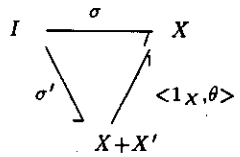
Para obter a nova transformación denominada *Narrowing con Abstracción* e representada por $\stackrel{na}{\Rightarrow}$ divídese novamente o proceso en tres fases :

i) a abstracción dunha parte do problema,

ii) a aplicación dunha forma de narrowing na parte previamente debilitada pola abstracción,

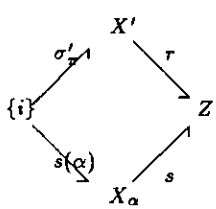
iii) a aplicación da 2-célula e incorporación das condicións demoradas.

i) Comparando este paso co anterior, agora só se realiza a abstracción de $\sigma_\pi = \sigma/\pi$ e a substitución σ é reemplazada por : $\sigma' = \sigma \circ j_X[\pi \leftarrow \sigma'_\pi \circ j_{X'}] : I \rightharpoonup X + X'$



Como no caso anterior, o codominio de $s(ex(\alpha))$ é $X + X_\alpha$.

ii) A segunda operación consiste na obtención do unificador máis xeral $\delta : X + X' + X_\alpha \rightharpoonup Z$ do problema $\sigma' \circ j_{X+X'} \stackrel{?}{=} s(ex(\alpha)) \circ j_{X+X_\alpha} : I \rightharpoonup X + X' + X_\alpha$. Se δ é tamén un coigualador de ese problema, pode expresarse igualmente como $<j_X, r, s> : X + X' + X_\alpha \rightharpoonup X + Z$, onde r e s se obteñen do *push-out* :



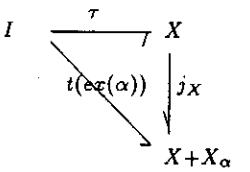
Igualmente ó caso anterior, existe unha inxección $Z \xrightarrow{j_Z} Z + Z' = X' + X_\alpha$.

iii) O último paso consiste na obtención do novo problema que contén como subproblemas :

a) ó par de substitucións obtido ó reemplazar σ por $t(ex(\alpha))$,

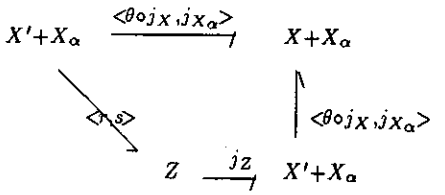
b) ás condicións demoradas pola abstracción (a substitución da abstracción, xunto co coigualador obtido).

a) Para conservar a compatibilidade nas variables, é preciso realizar o coproducto $X + X_\alpha$ xa que o novo problema contén ás variables previas (X) xunto coas variables (X_α) da 2-célula, tratándose polo tanto do par $\tau \circ j_X, t(ex(\alpha))$

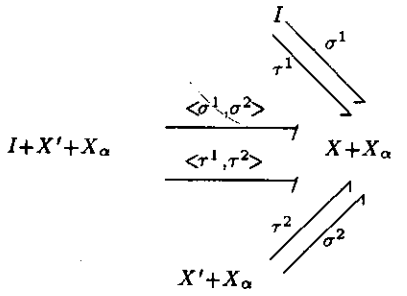


Xa que para toda posición π', $\pi' \perp \pi$, tense que $\sigma/\pi' = t(ex(\alpha))/\pi'$, esta primeira condición do novo problema pode expresarse como $\sigma^1 \stackrel{?}{=} \tau^1$, onde $\sigma^1 = \sigma \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}]$, $\tau^1 = \tau \circ j_X$.

b) A segunda condición, contén a expresión necesariamente demorada dado que non pode ser igualada sintácticamente (a diferencia do narrowing). Esta forma demorada é : $\sigma^2 \stackrel{?}{=} \tau^2$, onde $\tau^2 = < \theta \circ j_X, j_{X_\alpha} > : X' + X_\alpha \rightarrow X + X_\alpha$ e $\sigma^2 = <r, s> \circ j_Z \circ < \theta \circ j_X, j_{X_\alpha} >$.

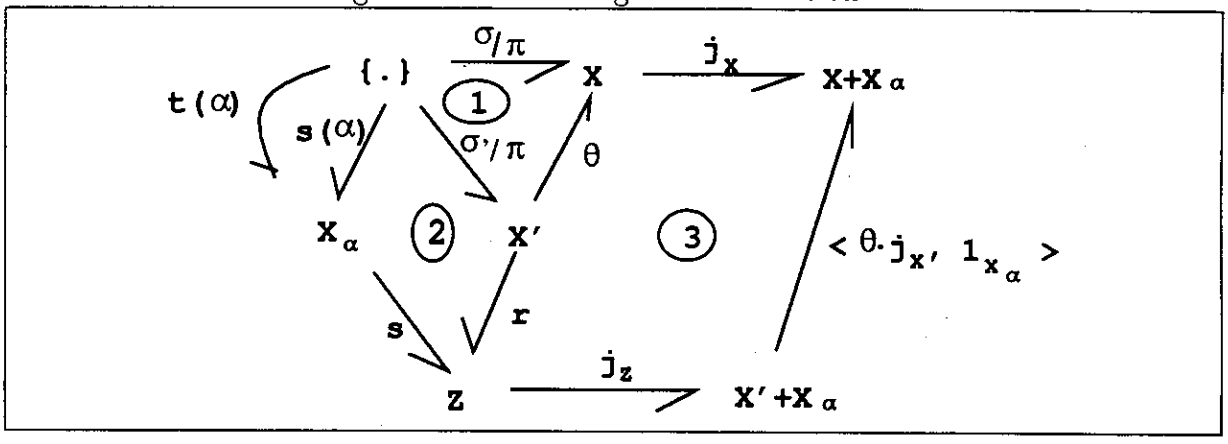


Por último, a reunión do par de subproblemas, produce como novo problema :



Na figura *4.3* móstrase o esquema xeral utilizado e na que :

Figura 4.3: Narrowing con Abstracción



- o diagrama representado por ① corresponde á abstracción,
- o correspondiente a ② é a extracción do unificador máis xeral,
- a operación empregada en ③ representa a inclusión das variables e o coproducto.

Neste caso, diráse que $\sigma \stackrel{?}{=} \tau$ é levado mediante *Narrowing con Abstracción* en $<\sigma^1,\sigma^2> \stackrel{?}{=} <\tau^1,\tau^2>$, o que será expresado por

$$\sigma \stackrel{?}{=} \tau \quad \stackrel{na}{\Rightarrow}_{\pi,\alpha} <\sigma^1,\sigma^2> \stackrel{?}{=} <\tau^1,\tau^2>$$

Tamén se usa $\stackrel{na}{\Rightarrow}$ sen explicitar a posición nen a 2-célula utilizada. Os unificadores que se obteñen polo novo método diránse obtidos por *narrowing con abstracción.*

Pode observarse unha certa semellanza co método que se segue de empregar a propiedade 4.4 cando a condición $\delta \in cU(..)$ é reemplazada por unha condición máis forte (coigualador), e se incorporan as novas condicións $\sigma^2 \stackrel{?}{=} \tau^2$.

Relaxed Paramodulation e Narrowing con Abstracción

Narrowing con Abstracción emprega unha condición máis forte que a de *Relaxed Paramodulation*, dado que unha das expresións a unificar non foi debilitada (*linearizada*).

4.6 *Sexan s,t dous termos. A condición top-unify(s,t) é equivalente á existencia do mgu(s',t'), onde s', t' son, respectivamente, a abstracción-posición de s e a de t.*

Dado que a condición *top-unify* equivale á existencia do unificador máis xeral da abstracción e do lado esquerdo da ecuación, coinciden cando se non hai variables repetidas neste. En cambio, se non existen variables repetidas no subtermo (e polo tanto a abstracción non é necesaria), a condición $mgu(s/\pi, l)$ é exactamente a condición necesaria para poder aplicar *narrowing*. Polo tanto, pode decirse que *narrowing con abstracción*, se atopa entre *Relaxed Paramodulation* e *narrowing*.

A linealidade das ecuacións (especialmente do lado esquerdo cando se atopan orientadas), é unha restricción usual [87, 190, 128] e acompañada doutras restriccións, garantiza o carácter canónico. Xustamente os casos de non linealidade, polo tanto os máis difíciles de tratar, é onde *narrowing con abstracción* mellora a *Relaxed Narrowing*. O eliminar un bó número de redundancias, faise máis difícil obter novos métodos sen perder a completitud. Así, modificacións equivalentes aplicadas a *Relaxed Narrowing* e a *narrowing con abstracción* eliminan neste unha menor proporción de solucións. Obsérvese para elo, a Tabla B.2 nos problemas penúltimo e antepenúltimo.

No seguinte exemplo vése diferencia entre a forma de actuación de *Lazy Paramodulation*, *Relaxed Paramodulation* e *Narrowing con Abstracción*.

Exemplo 4.17 $R = \{a \to b, f(a,b,x,x,b) \to g(a,b)\}$, $\Gamma = \{f(y,t,a,b,a) \stackrel{?}{=} g(y)\}$

- *a regra* $f(a,b,x,x,b) \to g(a,b)$ *pode ser aplicada en* $f(y,y,a,b,a)$ *por Lazy Paramodulation,*

- *Relaxed Paramodulation non permite esta aplicación, e debe usarse previamente a regra* $a \to b$ *na posición* 5 *e unha vez obtido o termo* $f(y,y,a,b,b)$ *xa pode ser aplicada* $f(a,b,x,x,b) \to g(a,b)$,

- *Narrowing con Abstracción non permite tampouco esta última transformación, ó non existir unificador do par* $\big(f(y_1,y_2,a,b,b), f(a,b,x,x,b)\big)$. *Para obter a solución debe ser usada novamente* $a \to b$ *na posición* 3 *producindo o termo* $f(y,y,b,b,b)$ *onde xa é posible empregar* $f(a,b,x,x,b) \to g(a,b)$ *obtendo* $\{g(a,b) \stackrel{?}{=} g(a,b),\ y \stackrel{?}{=} a, y \stackrel{?}{=} b\}$.

No Apéndice A móstrase unha implementación destes algoritmos. No Apéndice B obsérvase un estudio comparativo nunha serie de problemas, de exemplos resoltos cas mencionadas implementacións.

Pode observarse unha proximidade entre o *Narrowing con Abstracción* e a estratexia de *basic narrowing*. Nos dous casos, considérase en forma diferente as instanciacións das variables da regra ecuacional empregada e coas do problema dado. En certa medida pódese decir que corrixen a simetría das reglas previas, usando transformacións que se adaptan mellor ó contexto no que as transformacións son aplicadas. Amais, en ambas substitúense derivacións que actúan sobre subtermos do novo problema, por aplicacións de regras nos correspondentes subtermos previos.

4.3.5 Corrección e completitude do novo algoritmo

Próbase en primeiro lugar, a corrección da transformación $\overset{na}{\Rightarrow}$.

4.7 *Sexa* $\sigma \overset{?}{=} \tau \overset{na}{\Rightarrow}_{\pi,\alpha} <\sigma^1,\sigma^2> \overset{?}{=} <\tau^1,\tau^2>$ *e* λ *un E-unificador de* $<\sigma^1,\sigma^2> \overset{?}{=} <\tau^1,\tau^2>$. *Entón* $j_X \circ \lambda$ *é un E-unificador de* $\sigma \overset{?}{=} \tau$.

Proba. Por hipótese $\sigma^1\lambda =_E \tau^1\lambda$ ou sexa $\sigma \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}]\lambda =_E \tau \circ j_X\lambda$, así como $<\theta \circ j_X, j_{X_\alpha}> \lambda =_E <r,s> \circ j_Z \circ <\theta \circ j_X, j_{X_\alpha}> \lambda$. Compoñendo esta última igualdade con j_{X_α}, $j_{X_\alpha}\lambda = s \circ j_Z \circ <\theta \circ j_X, j_{X_\alpha}> \lambda$.

Próbase que $\sigma/\pi \circ j_X \circ \lambda =_E t(\alpha) \circ j_{X_\alpha} \circ \lambda$. Para elo, emprégase que $\sigma/\pi \circ j_X\lambda = t(\alpha) \circ j_{X_\alpha} \circ \lambda$. Esto é certo, posto que $\sigma/\pi \circ j_X \circ \lambda = (\sigma/\pi)'\theta \circ j_X \circ \lambda = (\sigma/\pi)' \circ j_{X'} \circ <\theta \circ j_X, j_{X_\alpha}> \lambda =_E (\sigma/\pi)' \circ j_{X'} \circ <r,s> \circ j_Z \circ <\theta \circ j_X, j_{X_\alpha}> \lambda = (\sigma/\pi)' \circ r \circ j_Z <\theta \circ j_X, j_Y> \lambda = s(\alpha) \circ s \circ j_Z <\theta \circ j_X, j_Y> \lambda =_E t(\alpha) \circ s \circ j_Z <\theta \circ j_X, j_{X_\alpha}> \lambda =_E t(\alpha) \circ j_{X_\alpha} \circ \lambda$.
De aquí se deduce a corrección, dado que $\sigma \circ j_X \circ \lambda = \sigma[\pi \leftarrow (\sigma/\pi)] \circ j_X \circ \lambda = \sigma \circ j_X \circ \lambda[\pi \leftarrow (\sigma/\pi) \circ j_X \circ \lambda] =_E \sigma \circ j_X \circ \lambda[\pi \leftarrow t(\alpha) \circ j_{X_\alpha} \circ \lambda] = \sigma \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}]\lambda =_E \tau \circ j_X \circ \lambda$ □.
Pode observarse que nesta proba téñense empregado as propiedades *2.3* e *2.8* das posicións e substitucións respectivamente, así como a monotonía e a congruencia da relación $=_E$.

4.8 *Sexa* $\lambda \in S(\sigma \overset{?}{=} \tau, \overset{na}{\Rightarrow})$, *entón* $\sigma\lambda =_E \tau\lambda$.

Esto resulta directamente da corrección de cada paso de $\overset{na}{\Rightarrow}$ pola propiedade anterior e da correción de cada paso de $\overset{mgu}{\Rightarrow}$ □.

Para probar a completitude do novo método, aprovéitase a completitude do narrowing usando ecuacións de $C(E)$, o que producirá como consecuencia a completitude de *Narrowing con Abstracción* usando estas mesmas ecuacións. Posteriormente se verá como se poden sustituir as secuencias que empregan os elementos de $C(E)$ que non se atopan no conxunto de ecuacións E dado.

4.9 *Sexa* $C(E)$ *o resultado de aplicar a complección de pares críticos a un conxunto* E *de 2-células.* $S(\sigma \overset{?}{=} \tau, \overset{na}{\Rightarrow})$ *é un conxunto completo de E-unificadores.*

Proba. Xa se probou que $C(E)$ é pechada, polo que, usando este conxunto de axiomas, $S(\sigma \overset{?}{=} \tau, \overset{n}{\Rightarrow})$ é completo. Por outra parte, calquera aplicación de narrowing é obtida como $\overset{na}{\Rightarrow}$ seguida da aplicación de $\overset{mgu}{\Rightarrow}$ na expresión indicada previamente como $\sigma^2 \overset{?}{=} \tau^2$. Polo tanto, $S(\sigma \overset{?}{=} \tau, \overset{na}{\Rightarrow})$ é un conxunto completo de C(E)-unificadores de $\sigma \overset{?}{=} \tau$ □.

Para reemplazar o uso dos pares críticos, precísase unha forma de eliminar operadores despois de obter a unificación das condicións demoradas.

Sexa $\sigma = <\sigma^1, \sigma^2>$, $\tau = <\tau^1, \tau^2>$. A transformación $\stackrel{d}{\Rightarrow}$ consiste na eliminación dos operadores seguida da factorización. Sexa $\sigma^2 = <\sigma_1, \ldots, \sigma_n>$, $\tau^2 = <\tau_1, \ldots, \tau_n>$ e $\lambda_f \bar{\sigma}_i$, $\lambda_f \bar{\tau}_i$ a factorización do primeiro operador (2.51) para σ_i e τ_i respectivamente. Usamos novamente a notación σ^2 para expresar $< sigma_1, \ldots, \sigma_{i-1}, \bar{\sigma}_i, \ldots, \sigma_n >$ (análogamente para τ^2). A continuación se definen $\bar{\sigma}, \bar{\tau}$ desde σ^2, τ^2 mediante : para cada x_i tal que $(\sigma^2)_{ep}/\pi_i^1 = x_i = (\sigma^2)_{ep}/\pi_i^k$ tense $\bar{\sigma}/\pi_i^1 = \sigma^2/\pi_i^1$ e $\bar{\sigma}/\pi_i^k = \tau^2/\pi_i^1, k \neq 1$ e $\bar{\tau} = \tau^2$. Seguidamente, en $\bar{\sigma}, \bar{\tau}$ emprégase novamente a eliminación de operadores, indicando o novo par tamén por $\bar{\sigma}, \bar{\tau}$. Con esta notación, a nova transformación $\stackrel{d}{\Rightarrow}$ ven dada por

$$\sigma \stackrel{?}{=} \tau \quad \stackrel{d}{\Rightarrow} <\sigma^1, \bar{\sigma}> \stackrel{?}{=} <\tau^1, \bar{\tau}>$$

A correción desta operación é unha consecuencia directa da monotonía da igualdade.

A continuación preséntase un resultado determinante para a completitude do novo método, xa que proba que os elementos de E^i poden ser sustituídos polos de E^{i-1}.

4.10 *Sexa $\sigma \stackrel{?}{=} \tau$ un problema de E-unificación e λ un E-unificador tal que $\lambda \in S(\sigma \stackrel{?}{=} \tau, \stackrel{ng}{\Rightarrow} \cup \stackrel{d}{\Rightarrow})$ con 2-células de $E \cup$ Pares-críticos(E). Entón $\lambda \in S(\sigma \stackrel{?}{=} \tau, \stackrel{ng}{\Rightarrow} \cup \stackrel{d}{\Rightarrow})$ con 2-células de E.*

Para simplificar a notación, non serán indicadas as inclusións j_X, j_{X_α} e a substitución $<\theta \circ j_X, 1_{X_\alpha}>$ exprésase como θ, mentras que a expresión $<r, s> \circ j_Z \circ <\theta \circ j_X, j_{X_\alpha}>$ será δ.

Proba. Aplicarase inducción no número n de pares críticos empregados.
Para $n = 0$ tódalas 2-células son de E e polo tanto o resultado é certo.
Suposto para $n - 1$. Sen pérdida de xeralidade, pode supoñerse que a primeira regra empregada é $\stackrel{ng}{\Rightarrow}$, aplicada na posición π de σ a un par crítico de $I \overset{\alpha}{\rightarrow} X_\alpha$ e $I \overset{\beta}{\rightarrow} X_\beta$. Este pode ter a seguinte forma e orientación :

1. $t(\alpha)\rho \stackrel{\gamma^{-1}}{\Rightarrow} s(\alpha)\rho[\omega \leftarrow t(\beta)\rho]$ ou ben :
2. $s(\alpha)\rho[\omega \leftarrow t(\beta)\rho] \stackrel{\gamma}{\Rightarrow} t(\alpha)\rho$

onde $\rho = mgu(s(\alpha)/\omega, s(\beta))$

Para o primeiro caso,

$$\sigma \stackrel{?}{=} \tau \stackrel{ng}{\Rightarrow}_{\pi,\gamma} \sigma_1 \stackrel{?}{=} \tau_1$$

onde $\sigma_1 = < \sigma[\pi \leftarrow s(\alpha)\rho[\omega \leftarrow t(\beta)\rho]], \delta_1 >$, $\tau_1 = < \tau, \theta_1 >$ sendo $\delta_1 = \big(mgu((\sigma/\pi)', t(\alpha)\rho)\big)\theta_1$, $\theta_1 = abst(\sigma/\pi)$ e a condición para que γ^{-1} sexa aplicada é a existencia do mgu sinalado.

Probarase que, por seleccións axeitadas, pode obterse o propio par $\sigma_1 \stackrel{?}{=} \tau_1$ empregando as regras α e β.

Para elo, aplícase $\stackrel{ng}{\Rightarrow}$ en σ, posición π coa 2-célula α^{-1} :

$$\sigma \stackrel{?}{=} \tau \stackrel{na}{\Rightarrow}_{\pi,\alpha^{-1}} \bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1$$

con $\bar{\sigma}_1 = < \sigma[\pi \leftarrow s(\alpha)], \delta_1' >$, $\bar{\tau}_1 = < \tau, \theta_1' >$, $\delta_1' = \Big(mgu((\sigma/\pi)', t(\alpha))\Big)\theta_1'$, $\theta_1' = \theta_1 = abst(\sigma/\pi)$. Pola separación de variables, a existencia de $mgu\Big((\sigma/\pi)', t(\alpha)\rho)\Big)$ implica a existencia de $mgu((\sigma/\pi)', t(\alpha))$ por *3.6.*

Na posición ω de $\sigma[\pi \leftarrow s(\alpha)]$ aplícase $\stackrel{na}{\Rightarrow}$ coa regra β de $\bar{\sigma}_1$, o que produce como novo problema :

$$\bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1 \stackrel{na}{\Rightarrow}_{\omega,\beta} \bar{\sigma}_2 \stackrel{?}{=} \bar{\tau}_2$$

onde $\bar{\sigma}_2 = < \sigma[\pi \leftarrow s(\alpha)][\pi \cdot \omega \leftarrow t(\beta)], \delta_1', \delta_2' >$, $\bar{\tau}_2 = < \tau, \theta_1, \theta_2' >$, con $\delta_2' = \Big(mgu((\sigma[\pi \leftarrow s(\alpha)'), s(\beta))\Big)\theta_1'$, $\theta_2' = abst(\sigma[\pi \leftarrow s(\alpha)])$. Esta operación pode realizarse, xa que pola propiedade 2 de *2.3*, $mgu\Big((\sigma[\pi \leftarrow s(\alpha)]/\pi \cdot \omega)', s(\beta)\Big) = mgu((s(\alpha)/\omega)', s(\beta)) \leq \rho$.

A continuación, pode aplicarse $\stackrel{mgu}{\Rightarrow}$ no par $\delta_1' \stackrel{?}{=} \theta_1'$ que proporciona ρ, como se deduce de utilizar a propiedade *4.4*. Aplicar ρ ó problema resultante e eliminar os operadores comúns mediante $\stackrel{d}{\Rightarrow}$ produce xustamente $\sigma_1 \stackrel{?}{=} \tau_1$.

No segundo caso, existen dúas posibilidades :

[a] que existan posicións ν, ν' (de $s(\alpha)$), tales que $\pi \cdot \nu$ é unha posición variable de σ, $\omega = \nu \cdot \nu'$ (esto é, ou ben $\pi \cdot \omega$ ou non existe ou é unha posición variable de σ),

[b] que a posición $\pi \cdot \omega$ sexa unha posición non variable de σ.

A aplicación de γ na posición π de σ, da lugar a :

$$\sigma \stackrel{?}{=} \tau \stackrel{na}{\Rightarrow}_{\pi,\gamma} \sigma_1 \stackrel{?}{=} \tau_1$$

onde $\sigma_1 = < \sigma[\pi \leftarrow t(\alpha)\rho], \delta_1 >$, $\tau_1 = < \tau, \theta_1 >$, $\delta_1 = \Big(mgu((\sigma/\pi)', s(\alpha)\rho[\omega \leftarrow t(\beta)\rho])\Big)\theta_1$, $\theta_1 = abst(\sigma/\pi)$ e por hipótese, existe este mgu.

Para o caso **[a]**, pode seleccionarse a posición π de σ e a regra α, de onde se obtén :

$$\sigma_0 \stackrel{?}{=} \tau_0 \stackrel{na}{\Rightarrow}_{\pi,\alpha} \bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1$$

onde $\bar{\sigma}_1 = < \sigma[\pi \leftarrow t(\alpha)], \delta_1' >, \bar{\tau}_1 = < \tau, \theta_1' >$, $\delta_1' = \Big(mgu((\sigma/\pi)', s(\alpha))\Big)\theta_1'$, $\theta_1' = abst(\sigma/\pi)$.

Esta selección é correcta, xa que a variable que se atopa na posición $\pi \cdot \nu$ ten sido renomeada e o $mgu((\sigma/\pi)', s(\alpha))$ é $mgu(\sigma/\pi \cdot \omega')', s(\alpha)/\omega') \cup mgu((\sigma/\pi \cdot \nu)', s(\alpha)/\cdot \nu)$, para toda $\omega' \in \Pi(s(\alpha)), \omega' \perp \nu$.

De feito, δ_1' e θ_1' pódese expresar como $< \delta_1'^1, \delta_1^2 >$ e $< \theta_1'^1, \theta_1'^2 >$ onde

- $\delta_1^1 = \Big(mgu((\sigma/\pi \cdot \nu)', s(\alpha)/\nu))\Big)\theta_1'^1 = s(\alpha)/\nu$, $\theta_1'^1 = abst(\sigma/\pi \cdot \nu)$,

- $\delta_1'^2 = \Big(mgu((\sigma/\pi\cdot\omega')', s(\alpha)/\omega'))\Big)\theta_1'^2,\ \theta_1^2 = abst(\sigma/\pi\cdot\omega'),\ \omega' \perp \nu$

Aplicando a regra β na posición ν' de δ_1^1 (que non é variable por definición de par crítico), obtense o novo problema :

$$\bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1 \ \stackrel{na}{\Rightarrow}_{\beta,\nu'}\ \bar{\sigma}_2 \stackrel{?}{=} \bar{\tau}_2$$

onde $\bar{\sigma}_2 = < \sigma[\pi \leftarrow t(\alpha)], \delta_1'^1[\nu' \leftarrow t(\beta)], \delta_1^2, \delta_2 >$, $\bar{\tau}_2 = < \tau, \theta_1'^1, \theta_1^2, \theta_2 >$ con $\delta_2 = \Big(mgu((s(\alpha)/\omega)', s(\beta))\Big)\theta_1$, $\theta_2 = abst(s(\alpha)/\omega)$.

A transformación $\stackrel{mgu}{\Rightarrow}$ aplicada ó par $\delta_1 \stackrel{?}{=} \theta_1$ proporciona ρ, o cal, aplicado ó resto do problema produce

$$\bar{\sigma}_2 \stackrel{?}{=} \bar{\tau}_2 \ \stackrel{mgu}{\Rightarrow}\ \bar{\sigma}_3 \stackrel{?}{=} \bar{\tau}_3$$

para $\bar{\sigma}_3 = <\sigma[\pi \leftarrow t(\alpha)\rho], s(\alpha)\rho/\nu[\nu' \leftarrow t(\beta)\rho], \delta_1'^2\rho>$, $\bar{\tau}_3 = <\tau, \theta_1'^1, \theta_1^2>$

O par $<s(\alpha/\nu)\rho[\nu' \leftarrow t(\beta)\rho], \delta_1^2\rho> \stackrel{?}{=} <x, \theta_1^2>$ pode expresarse logo da eliminación dos operadores comúns por $\stackrel{d}{\Rightarrow}$, (pola asociatividade do mgu e xa que as posicións $\varpi \in \Pi(s(\alpha))$, $\varpi < \nu$ non son variables e coinciden coas posicións $\sigma/\pi\varpi$), como $\delta_1 \stackrel{?}{=} \theta_1$ de onde se obtén o mesmo problema resultado de aplicar o par crítico γ.

Para reemplazar a secuencia derivada de [b] selecciónase en primeiro lugar a posición $\pi\cdot\omega$ de σ onde é aplicada β^{-1}. Esto produce :

$$\sigma \stackrel{?}{=} \tau \ \stackrel{na}{\Rightarrow} \bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1$$

con $\bar{\sigma}_1 = < \sigma[\pi\cdot\omega \leftarrow s(\beta)], \delta_1' >$, $\bar{\tau}_1 = < \tau, \theta_1' >$, $\delta_1' = \Big(mgu((\sigma/\pi\cdot\omega)', t(\beta))\Big)\theta_1'$, $\theta_1' = abst(\sigma/\pi\cdot\omega)$. A existencia do *mgu* está garantida, xa que é a su vez unha condición necesaria para a existencia do $mgu((\sigma/\pi)', s(\alpha)[\omega \leftarrow t(\beta)])$ unificador existente, xa que $dom(\rho) \cap X = \emptyset$ por *3.5.*

Seleccionando $\sigma[\pi\cdot\omega \leftarrow s(\beta)]$, posición π e regra α, tense a transformación :

$$\bar{\sigma}_1 \stackrel{?}{=} \bar{\tau}_1 \ \stackrel{na}{\Rightarrow} \bar{\sigma}_2 \stackrel{?}{=} \bar{\tau}_2$$

onde $\bar{\sigma}_2 = <\sigma[\pi\cdot\omega \leftarrow s(\beta)][\pi \leftarrow t(\alpha)], \delta_1', \delta_2>$, $\bar{\tau}_2 = <\tau, \theta_1', \theta_2>$,$\delta_2 = \Big(mgu((\sigma[\pi\cdot\omega \leftarrow s(\beta)]/\pi)', s(\alpha))\Big)\theta_2$, $\theta_2 = abst(\sigma[\pi\cdot\omega \leftarrow s(\beta)]/\pi)$. A existencia de δ_2 está asegurada, posto que :

- $s(\alpha)\rho/\omega = s(\beta)\rho$ e polo tanto $s(\alpha)[\pi \leftarrow s(\beta)]\rho = s(\alpha)\rho$,
- se existe $mgu(s,t)$, tamén existe $mgu(s[\pi \leftarrow u], t[\pi \leftarrow u])$,
- xa que $(\sigma[\pi\cdot\omega \leftarrow s(\beta)])'/\pi \leq \sigma[\pi\cdot\omega \leftarrow s(\beta)\rho]$ e pola separación de variables de β e α, se ten a existencia do *mgu* requerido.

Aquí o problema subdivídese en :

1. cando $s(\alpha)/\omega$ ten variables comúns con $s(\alpha)/\omega'$, para toda $\omega' \in \Pi(s(\alpha)), \omega' \perp \omega$ e estas variables son instanciadas por $\bar{\delta}_2$,

2. noutro caso.

Para 1 : Neste caso, pode existir algunha variable instanciada por δ_2 que pertenece a $cod(\rho)$. Nótese que δ_2 pode instanciar un certo número de variables de $s(\alpha), s(\beta)$, eliminadas por ρ.

Probarase que, se o unificador máis xeneral ten sido expresado nunha forma factorizada, ρ pode ser obtido igualmente empregando $\stackrel{mgu}{\Rightarrow}$ nas ecuacións de $<\delta_1', \delta_2> \stackrel{?}{=} <\theta_1', \theta_2>$. Obtido ρ, e aplicado nas novas ecuacións, obtense o propio $\delta_1 \stackrel{?}{=} \theta_1$.

Para elo, empregarase a factorización do unificador ρ. Sexan μ, ν substitucións e η un unificador sintáctico deste par. Defínese o *grado de factorización dunha variable* x en (μ, ν, η) por

- 0 se $x\eta = x$,
- i, se $i - 1 = max\Big\{grado\_factorizacion(y)|$ existe $\pi \in \Pi_x\mu,\ y \in cod((\nu/\pi)_{ep})\Big\} \cup \Big\{(grado\_factorizacion(y)|$ existe $\pi \in \Pi_x\nu,\ y \in cod((\mu/\pi)_{ep})\Big\}$.

Supónse un orden nas variables para orientar as instancias mutuas dos pares de variables que se atopan na mesma clase para a relación *válida* que se establece en $\mu \stackrel{?}{=} \nu$.

O grado de factorización i de (μ, ν, η) é o máximo dos grados de factorización das variables de $cod(\eta)$, de forma que pode supoñerse a existencia dunha substitución $\bar{\eta}$ tal que $\eta = \bar{\eta}\ldots\bar{\eta}$ un número i de veces. Sexan : x unha variable tal que $grado\_factorizacion(x) = i$, π unha posición $\pi \in \Pi_x(\mu) \cap \Pi(\nu)$ con $max\{grado\_factorizacion(y \in cod(\nu/\pi))\} = i - 1$. Esto permite determiñar a instanciación $\bar{\eta}_{\{x\}} = \nu/\pi$.

Para probar o resultado, aplícase inducción no grado de factorización i de $(s(\alpha)/\omega, s(\beta), \rho)$.

Para $i = 0$, esto significa que $\rho = 1_{X_\alpha + X_\beta}$ y polo tanto $\delta_1 = <\delta_1', \delta_2> = \rho\delta_1$ (dado que non se xeran ecuacións entre $s(\alpha)/\omega$ y $s(\beta)$).

Suposto para $i - 1$. Posto que o unificador máis xeral se expresa en forma factorizada, que en $\delta_2 \stackrel{?}{=} \theta_2$, se $x_\beta \in cod(s(\beta))$ e $grado\_factorizacion(x_\beta)$ en $(s(\alpha)/\omega, s(\beta), \rho) = i$, existe $\omega_{x_\beta} \in \Pi(s(\alpha)$, e unha variable x'_β, $x'_\beta\theta_2 = x_\beta$ tal que $x'_\beta\delta_2 = s(\alpha)/\omega_{x_\beta}$ e o grado de factorización das variables de $s(\alpha)/\omega_{x_\beta}$ é $< i$. Análogamente, se x_α ten $grado\_factorizacion = i$, existe $\omega_{x_\alpha}, x'_\alpha$, $x'_\alpha\theta_2 = x_\alpha$, $x'_\alpha\delta_2 = s(\beta)/\omega_{x_\alpha}$.

Suposto que para toda variable $x \in dom(\rho)$, tal que $grado\_factorizacion(x) < i$, a partir de $\delta_2 \stackrel{?}{=} \theta_2$ se obtén aplicando a transformación $\stackrel{mgu}{\Rightarrow}$ unha ecuación que, por abuso de notación, pode expresarse como $x\delta \stackrel{?}{=} x\rho$ e esta variable tén sido eliminada no resto das ecuacións.

Sexa y, $grado\_factorizacion(y) = i$. Xa que $y\rho$ pode expresarse como $y'\delta_2\rho$ onde y' corresponde a unha variable resultante de abstraer y, que o grado de factorización das variables de $y'\delta_2$ é $< i$, a eliminación de variables ($\overset{mgu}{\Rightarrow}$), produce a ecuación $y \overset{?}{=} \delta_2\rho$ sendo esta última expresión $=y\rho$. Obtido ρ, as novas ecuacións resultantes de $<\delta_1', \delta_2> \overset{?}{=} <\theta_1', \theta_2> \overset{mgu}{\Rightarrow}_\rho \overset{d}{\Rightarrow}^* \delta \overset{?}{=} \theta$, coinciden con $\theta_1 \overset{?}{=} \delta_1$. A operación $\overset{d}{\Rightarrow}$ precísase para eliminar os operadores nas ecuacións resultado de aplicar ρ (dado que o unificador máis xeral poidera non obterse).

2. O par $\delta_2 \overset{?}{=} \theta_2$ pode expresarse como $<\delta_2^1, \delta_2^2> \overset{?}{=} <\theta_2^1, \theta_2^2>$ onde

- $\theta_2^1 = abst(s(\beta))$, $\theta_2^2 = abst(\sigma/\pi \cdot \omega')$, $\omega' \perp \omega$,
- $\delta_2^1 = \Big(mgu((s(\beta))', s(\alpha)/\omega)\Big)\theta_2^1$, $\delta_2^2 = \Big(mgu((\sigma/\pi \cdot \omega)', s(\alpha)/\omega')\Big)\theta_2^2$.

Esto é certo, dado que as posicións $\leq \pi \cdot \omega$ de σ non son variables. Aplicando $\overset{mgu}{\Rightarrow}$ no par $\delta_2^1 \overset{?}{=} \theta_2^1$ obtense ρ, que aplicado no resto do problema e posterior uso de $\overset{d}{\Rightarrow}$ produce :

$$\bar{\sigma}_2 \overset{?}{=} \bar{\tau}_2 \overset{mgu}{\Rightarrow} \overset{d}{\Rightarrow}^* \bar{\sigma}_3 \overset{?}{=} \bar{\tau}_3$$

onde $\bar{\sigma}_3 = <\sigma[\pi \cdot \omega \leftarrow s(\beta)\rho][\pi \leftarrow t(\alpha)], \delta_1'\rho, \delta_2^2>$, $\bar{\tau}_3 = <\tau, \theta_1', \theta_2^2>$. Pero $\sigma[\pi \cdot \omega \leftarrow s(\beta)\rho][\pi \leftarrow t(\alpha)\rho] = \sigma[\pi \leftarrow t(\alpha)\rho]$ e, xa que $(s(\alpha)/\omega)\rho = s(\beta)\rho$, se deduce a igualdade $\Big\langle mgu((\sigma/\pi \cdot \omega)', t(\beta)\rho), mgu\Big((\sigma/\pi \cdot \omega')', s(\alpha)\rho/\omega'\Big)\Big\rangle = mgu\Big((\sigma/\pi)', s(\alpha)\rho[\omega \leftarrow t(\beta)\rho]\Big)$. Polo tanto, $\bar{\sigma}_3 \overset{?}{=} \bar{\tau}_3$ coincide con $\sigma_1 \overset{?}{=} \tau_1$. □.

4.11 *Sexa E un conxunto de ecuacións expresado como 2-células. A aplicación de $\overset{na}{\Rightarrow}$, $\overset{mgu}{\Rightarrow}$ e $\overset{d}{\Rightarrow}$ a un problema de unificación Γ, proporciona un conxunto completo de E-unificadores de Γ.*

Proba. En primeiro lugar, constrúese o conxunto pechado $C(E)$ de 2-células obtido dende E. Pola propiedade *4.3*, $S(\Gamma, \overset{n}{\Rightarrow})$ proporciona un conxunto completo de E-unificadores de Γ se se empregan para elo 2-células de $C(E)$. Por outra parte, toda aplicación de $\overset{n}{\Rightarrow}$ pode ser sustituída por unha secuencia de $\overset{na}{\Rightarrow}$ e $\overset{mgu}{\Rightarrow}$ (aplicando esta última na parte $\delta \overset{?}{=} \theta$). Sexa λ un E-unificador de Γ. Existe polo tanto unha secuencia de $\overset{na}{\Rightarrow}$ e de $\overset{mgu}{\Rightarrow}$ que proporciona $\mu \leq_E \lambda$. Xa que existe un $i \geq 0$ tal que para toda 2-célula γ de $C(E)$ empregada na secuencia que produce μ, verifícase que $\gamma \in E^i$, e aplicando reiteradamente a anterior propiedade, obtense o resultado □.

Incorporación de condicións

A transformación $\overset{na}{\Rightarrow}$ será modificada para incluír ecuacións condicionais.

A aplicación da 2-célula condicional α dada por $\Big(I_\alpha \overset{\overset{s(\alpha)}{t(\alpha)}}{\Rightarrow} X_\alpha,\ I_\alpha^c \overset{c_\alpha}{\underset{c_\alpha}{\Rightarrow}} X_\alpha + X_\alpha^e\Big)$, realízase en igual forma ó caso anterior, modificando os novos unificandos mediante a incorporación da expresión condicional e incluíndo o novo conxunto de variables X_α^e.

Nos diagramas previos para $\overset{nca}{\Longrightarrow}$, simplemente se incorpora (á dereita) o novo conxunto $X + X_\alpha + X^e_\alpha$.

Esta nova transformación actúa da seguinte forma :

$$\sigma \overset{?}{=} \tau \quad \overset{nca}{\Longrightarrow}_{\pi,\alpha} < \sigma_1, \delta, \kappa > \overset{?}{=} < \tau_1, \theta, \xi > \quad \text{onde}$$

- $\sigma_1 = \sigma \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}]$, $\tau_1 = \tau \circ j_X$,
- $\delta = \Big(mgu\big((\sigma/\pi)', s(\alpha)\big)\Big)\theta$, $\theta = abst(\sigma/\pi)$,
- $\kappa = c_\alpha \circ j_{X^e_\alpha}$, $\xi = c'_\alpha \circ j_{X^e_\alpha}$

O novo problema é polo tanto un par de substitucións entre os conxuntos $I + X' + X_\alpha + X^e_\alpha$ e $X + X_\alpha + X^e_\alpha$.

4.12 *Se* $\lambda \in S(\sigma \overset{?}{=} \tau, \overset{nca}{\Longrightarrow})$ *entón* $\lambda \in Unif_E(\sigma \overset{?}{=} \tau)$.

Proba. Aplícase inducción en n onde n é o número de pasos de $\overset{nca}{\Longrightarrow}$ empregados. Para $n = 0$, $\lambda = mgu(\sigma, \lambda)$.

Suposto para $n - 1$. Sexa $\sigma \overset{?}{=} \tau \quad \overset{nca}{\Longrightarrow} < \sigma_1, \delta, \kappa > \overset{?}{=} < \tau_1, \theta, \xi >$, $\bar{\lambda}$ tal que $\bar{\lambda} \in S(< \sigma_1, \delta, \kappa > \overset{?}{=} < \tau_1, \theta, \xi >, \overset{nca}{\Longrightarrow})$, e $\lambda = j_X \circ \bar{\lambda}$. Usando a notación anterior, por inducción $c_\alpha \circ j_{X_\alpha + X^e_\alpha} \bar{\lambda} =_E c_\alpha \circ j_{X_\alpha + X^e_\alpha} \bar{\lambda}$ y por conseguinte $s(\alpha) \circ j_{X_\alpha} \circ \bar{\lambda} =_E t(\alpha) \circ j_{X_\alpha} \circ \bar{\lambda}$. O resto é semellante á propiedade 4.7 □.

Para probar a completitude de $\overset{nca}{\Longrightarrow}$ emprégase o mesmo esquema que o utilizado para o caso sen condicións.

4.13 *Sexa C(E) o resultado de aplicar a complección por pares críticos a un conxunto E de 2-células condicionais. Entón* $S(\sigma \overset{?}{=} \tau, \overset{nca}{\Longrightarrow})$ *usando* $C(E)$ *é un conxunto completo de E-unificadores de* $\sigma \overset{?}{=} \tau$.

Proba. A aplicación de $\overset{ncd}{\Rightarrow}$ pode ser sustituída por $\overset{nca}{\Longrightarrow}$ seguida da transformación $\overset{mgu}{\Rightarrow}$ en $\delta \overset{?}{=} \theta$ (usando a notación anterior). Da completitude de $\overset{ncd}{\Rightarrow}$ dedúcese o resultado □.

4.14 *Sexa* $\sigma \overset{?}{=} \tau$ *un problema de E-unificación do que* λ *é unha solución obtida aplicando* $\overset{nca}{\Longrightarrow}$, $\overset{mgu}{\Rightarrow}$, $\overset{d}{\Rightarrow}$ *e 2-células de* $E \cup$ *Pares-críticos*(E). *Entón* $\lambda \in S(\sigma \overset{?}{=} \tau, \overset{nca}{\Longrightarrow} \cup \overset{d}{\Rightarrow})$ *con 2-células de E.*

Proba. A demostración é semellante á do caso sen condicións 4.10. Usando a mesma notación que en aquel caso, o par crítico xerado pode tratarse de :

$$\Big(\{i\} \underset{t(\gamma)}{\overset{s(\gamma)}{\Downarrow \gamma}} X_\gamma,\ I^c_\alpha + I^c_\beta \overset{c_\alpha}{\underset{c_\beta}{\rightarrow}} X_\gamma + X^e_\gamma\Big)$$

ou o seu inverso, γ^{-1} que difire de γ exclusivamente en que os respectivos inicio e destino, $s(\gamma)$ e $t(\gamma)$, teñen sido intercambiados.

Lémbrase que a aplicación de β poido producirse tanto na parte non condicional $s(\alpha)$ como na condicional c_α, c'_α. As respectivas substitucións son :

- $s(\gamma) = s(\alpha) \circ j_{X_\alpha} \circ \rho[\omega \leftarrow t(\beta) \circ j_{X_\beta} \circ \rho]$,
- $t(\gamma) = t(\alpha) \circ j_{X_\alpha} \circ \rho$,
- $c_\gamma = < c_\alpha \circ j_{X_\alpha + X^e_\alpha} < \rho \circ j_{X_\alpha}, j_{X^e_\alpha}, c_\beta \circ j_{X_\beta + X^e_\beta} < \rho \circ j_{X_\alpha}, j_{X^e_\beta} >>$,
- $c'_\gamma = < c'_\alpha \circ j_{X_\alpha + X^e_\alpha} < \rho \circ j_{X_\alpha}, j_{X^e_\alpha}, c'_\beta \circ j_{X_\beta + X^e_\beta} < \rho \circ j_{X_\alpha}, j_{X^e_\beta} >>$

Dado que $\rho = mgu(s(\alpha)/\omega, s(\beta))$ pode ser obtido mediante a aplicación de $\overset{nca}{\Longrightarrow}$ coas 2-células α, β (ou as súas inversas), a 2-célula γ (respectivamente γ^{-1}) é imitada. Posto que cada un dos *Pares críticos(E)* pode ser sustituído, obtense unha secuencia que produce λ. No caso de que γ se forme por aplicación de β na parte condicional, dado que γ foi aplicada, dedúcese que α pode ser igualmente aplicada (pola separación das variables e a propiedade *3.6*). Introducidas as expresións condicinais $c_\alpha \overset{?}{=} c'_\alpha$, pode aplicarse β na correspondente posición e $\rho = mgu(c_\alpha/\omega, s(\beta))$ é obtido empregando a transformación $\overset{mgu}{\Rightarrow}$, que, tras aplicar $\overset{d}{\Rightarrow}$ para eliminar os operadores, da o mesmo resultado que logo de aplicar γ □.

4.15 *Sexa E un conxunto de ecuacións condicionais, expresado como 2-células. O conxunto $S(\Gamma, \overset{nca}{\Longrightarrow} \cup \overset{d}{\Rightarrow})$, proporciona un conxunto completo de E-unificadores de Γ.*

Proba. Esta é equivalente ó caso non condicional 4.11. Xa que todo E-unificador de $\sigma \overset{?}{=} \tau$ pode ser obtido dende narrowing condicional usando 2-células de $C(E)$, existe unha serie de transformacións de $\overset{nca}{\Longrightarrow}$, $\overset{mgu}{\Rightarrow}$ que proporcionan λ usando as mesmas 2-células (aplicando directamente $\overset{mgu}{\Rightarrow}$ nas expresións $\delta \overset{?}{=} \theta$). Posto que toda 2-célula de $C(E)$ éo tamén dunha E^i, existe j, tal que as 2-células empregadas sonno de E^j. Aplicando reiteradamente a proposición anterior, obtense o resultado. □.

4.3.6 Melloras no novo algoritmo

Pode observarse que na aplicación de $\overset{ng}{\Rightarrow}$ (resp. $\overset{nca}{\Longrightarrow}$) para obter as solucións do problema inicial $\sigma \overset{?}{=} \tau$, e dadas as solucións do novo problema $\sigma_1 \overset{?}{=} \tau_1$, é preciso compoñer éstas coa inclusión j_X, dado que só son admitidas como solucións aquelas substitucións que modifican exclusivamente as variables do problema inicial. Pero en moitos problemas, existen diferentes instanciacións de variables dos axiomas sen relevancia nun problema dado. Por elo, se o novo problema contén subproblemas cuias solucións non poden modificar o problema inicial, estes subproblemas poden ser resoltos esixindo exclusivamente a obtención da súa resolubilidade. Esta situación darase cunha certa frecuencia ó incorporar ecuacións condicionais.

Exemplo 4.18 $R = \{f(x,a) \stackrel{?}{=} f(b,y) \stackrel{\alpha}{\Rightarrow} a \to b, \quad \stackrel{\beta}{\Rightarrow} f(b,y) \to f(x,a)\}, \quad \Gamma = \{a \stackrel{?}{=} b\}$.
A resolubilidade de $f(x,a) \stackrel{?}{=} f(b,y)$ *é inmediata por ter unificador máis xeral, o que evita ter que empregar a regra* β.

Respecto das estratexias de *narrowing* que poden ser incorporadas a *Narrowing con Abstracción*, encóntrase que baixo certas condicións se poden incluír a conmutación e a eliminación de regras en posicións instanciación de variables das 2-células.

4.16 *Sexa* $\sigma \stackrel{?}{=} \tau$ *un problema de E- unificación, e sexa* λ *un E-unificador de* $\sigma \stackrel{?}{=} \tau$ *obtido a partir da secuencia :*

$$\sigma \stackrel{?}{=} \tau \stackrel{nca}{\Longrightarrow}_{\pi,\alpha} < \sigma_1, \delta_1, c_\alpha > \stackrel{?}{=} < \tau_1, \theta_1, c'_\alpha > \stackrel{nca}{\Longrightarrow}_{\pi',\beta} < \sigma_2, \delta_1, c_\alpha, \delta_2, c_\beta > \stackrel{?}{=} < \tau_2, \theta_1, c'_\alpha, \theta_2, c'_\beta >$$

Se a posición π' *de* σ_1 *na que é aplicada, é independente de* π, *existe outra secuencia que produce* λ.

Proba. Xa que $\pi \perp \pi'$, entón $\sigma[\pi \leftarrow t(\alpha)][\pi' \leftarrow t(\beta)] = \sigma[\pi' \leftarrow t(\beta)][\pi \leftarrow t(\alpha)]$ pola conmutatividade. As correspondentes inclusións e coproductos tamén verifican a conmutatividade. □.

Este caso é semellante ó cuarto caso do lema de intercambio de Moser[129].

Exemplo 4.19 $R = \{f(x,x) \stackrel{\alpha}{\to} k(x,x),\ g(a) \stackrel{\beta}{\to} a,\ h(b) \stackrel{\gamma}{\to} a\}$, $\Gamma = \{f(g(x),h(y)) \stackrel{?}{=} k(x,x)\}$.
A secuencia : $\Gamma \stackrel{na}{\Rightarrow}_{(y/b)} f(g(x),a) \stackrel{?}{=} k(x,x) \stackrel{na}{\Rightarrow}_{(x/a)} f(a,a) \stackrel{?}{=} k(a,a)$ *pode reemplazarse usando unha selección left-to-right por :* $\Gamma \stackrel{na}{\Rightarrow}_{(x/a)} f(a,h(y)) \stackrel{na}{\Rightarrow}_{(y/b)} f(a,a) \stackrel{?}{=} k(a,a)$.

Se nos métodos *down-up* emprégase implícitamente a completitude de *narrowing* coa incorporación dos pares críticos xerados por complección sen fallo, e como se indicou na nota correspondente á propiedade *4.3* tamén *basic narrowing* é completo. Deste xeito, poden eliminarse certas ramas da árbore, examinando os pares críticos xerados e comprobando que tal secuencia non se corresponde cunha derivación de *basic narrowing*.

Exemplo 4.20 $R = \{f(a,b,x) \to h(a,b,x),\ a \to b\}$, $\Gamma = \{f(x,x,a) \stackrel{?}{=} h(x,x,b)\}$.
A secuencia seguinte $\Gamma \stackrel{na}{\Rightarrow} \{h(a,b,y) \stackrel{?}{=} h(b,b,b),\ x \stackrel{?}{=} a,\ x \stackrel{?}{=} b, y \stackrel{?}{=} a\} \stackrel{na}{\Rightarrow} \{h(a,b,y) \stackrel{?}{=} h(b,b,b),\ x \stackrel{?}{=} b,\ x \stackrel{?}{=} b, y \stackrel{?}{=} a\} \stackrel{mgu}{\Rightarrow} \{h(a,b,a) \stackrel{?}{=} h(b,b,b)\} \stackrel{na}{\Rightarrow} \{h(a,b,b) \stackrel{?}{=} h(a,b,b)\}$ *pode ser eliminada : esta secuencia convírtese en* $\Gamma \stackrel{n}{\Rightarrow} \{h(a,b,a) \stackrel{?}{=} h(b,b,b)\} \stackrel{n}{\Rightarrow} \{h(a,b,b) \stackrel{?}{=} h(a,b,b)\}$ *que non é básica, xa que se modifica a posición da variable* y.

Notese que, unha vez coñecida a secuencia de *Narrowing con Abstracción*, os pares críticos e a secuencia de *narrowing* correspondente se obteñen a partir das 2-células empregadas nas respectivas partes demoradas, como se mostra na propiedade 4.10.

Esta mellora suxire un plan máis radical para eliminar solucións redundantes, dado que posiblemente moitas estratexias de narrowing, completas en teorías canónicas, sonno tamén ó incorporar as ecuacións xeradas pola complección por pares críticos.

Outra forma de incorporar as estratexias do tipo de *basic narrowing* consiste en impoñer restriccións nas posicións que previamente corresponderon a posicións de variables dos axiomas empregados esixindo que sexan posicións básicas. Sen embargo, esta estratexia non é completa como se mostra a continuación :

Exemplo 4.21 $R = \{f(gx,x) \to h(x,x),\ h(a,b) \to k(a,b),\ l(gx,x) \to k(x,x),\ a \to b\}$, $\Gamma = \{f(ga,a) \stackrel{?}{=} l(ga,a)\}$. *A única forma de obter o E-unificador* ε *é pola secuencia* $\{f(ga,a) \stackrel{?}{=} l(ga,a)\} \stackrel{na}{\Rightarrow} \{h(x,x) \stackrel{?}{=} l(ga,a), x \stackrel{?}{=} a\}$, $\stackrel{mgu}{\Rightarrow} \{h(a,a) \stackrel{?}{=} l(ga,a)\}$ $\stackrel{na}{\Rightarrow} \{h(a,b) \stackrel{?}{=} l(ga,a)\}$ $\stackrel{na}{\Rightarrow} \{k(a,b) \stackrel{?}{=} l(ga,a)\}$ $\stackrel{na}{\Rightarrow} \{k(a,a) \stackrel{?}{=} l(ga,a)\}$ $\stackrel{na}{\Rightarrow} \{l(ga,a) \stackrel{?}{=} l(ga,a)\}$ *neste senso ou no seu oposto. A etapa* $h(a,a) \to h(a,b)$ *non é básica, nen o é* $k(a,a) \to k(a,b)$.

Este exemplo proba que as variables repetidas na súa parte dereita non poden bloquear as aplicacións de ecuacións. O problema aparece novamente se se considera unha variable repetida na parte esquerda da regra, xa que nunha etapa previa pode aparecer algunha expresión da forma $x \stackrel{?}{=} t[x_\alpha, x_\alpha]$, $x \stackrel{?}{=} t'$ e onde se precise aplicar outras ecuacións para resolver $t[x_\alpha, x_\alpha] \stackrel{?}{=} t'$, polo que as posicións da variable x_α teñen que ser abstraídas (ou ben se se produce antes unha instanciación debe permitirse aplicar regras en tales posicións). Esto mostra que só baixo condicións moi fortes de linealidade, pódese agardar, sen reconstruír a secuencia de narrowing, a completitude destas restriccións.

O problema da non orientación en teorías xerais difícilmente ten unha solución suficientemente xeral que sexa completamente satisfactoria. Como se ten visto, no caso xeral se só empregan as ecuacións nun senso único, pérdese a completitude, así como se se empregan nunha parte do problema (termo do par ou neste caso nunha substitución do par). Pero elo ten como consecuencia un aumento espectacular do número de solucións redundantes. Esto pode ser evitado en parte mediante mecanismos que impidan localmente o uso dunha de ambas orientacións da ecuación. Unha forma de resolver o problema é o de recurrir ó procedemento de reconstrucción da secuencia de narrowing, e aplicar que toda derivación é orientada nun ou noutro senso, pola propiedade *2.40*. Desta forma, as solucións redundantes son detectadas ó achar unha derivación que non foi dirixida por $\succ$. Esto pode observarse en :

Exemplo 4.22 $R = \{f(h(x,y),k(x)) \stackrel{\alpha}{\Rightarrow} g(kx,ky),\ h(x,x) \stackrel{\beta}{\Rightarrow} kx\}$, $\Gamma = \{f(z,z) \stackrel{?}{=} g(kz,kz)\}$. *Supóñase que, para este caso, o orden empregado na propiedade 2.40 é* $f > g$. *Entre as derivacións* $\Gamma \stackrel{na}{\Rightarrow}_\alpha \{g(kx,ky) \stackrel{?}{=} g(kz,kz),\ z \stackrel{?}{=} h(x,y), z \stackrel{?}{=} k(x)\} \Rightarrow_\beta$

$\{g(kx,kx) \stackrel{?}{=} g(kx,kx),\ z \stackrel{?}{=} kx, y \stackrel{?}{=} x\}$ *e*
$\Gamma \stackrel{na}{\Rightarrow}_{\alpha^{-1}} \{f(h(x,y),kx) \stackrel{?}{=} f(z,z),\ z \stackrel{?}{=} kx, z \stackrel{?}{=} ky\} \Rightarrow_{\beta} \{f(kx,kx) \stackrel{?}{=} f(kx,kx),\ z \stackrel{?}{=} kx, x \stackrel{?}{=} y\}$, *a primeira das derivacións corresponde o uso do par crítico* $f(kx,kx) \stackrel{\gamma}{\Rightarrow} g(kx,kx)$ *e* γ^{-1} *no segundo. Polo orden seleccionado, a segunda derivación non tén o mesmo senso que o inducido por* $\succ$ *e polo tanto pode ser retirada.*

Nos algoritmos de unificación que representan os termos por grafos[137], utilízanse as relacións de equivalencia *válidas*. Esto permite empregar un único representante para toda clase de equivalencia (e a instanciación é conxunta para toda ella). A continuación danse as nocións de subtermos conectados, para indicar que se atopan na mesma clase para a relación válida que se establece na unificación de s e t.

Definición 4.11 *Sexan* $\sigma, \tau : I \rightharpoonup X$ *substitucións e* π *unha posición de* σ. *A posición* π' *(de* σ *ou de* τ*) está* conectada *con* π *(o que se expresa por* $\pi \parallel \pi'$*) se se verifican algunha das condicións:*

- $\pi = \pi'$ *e* π' *é posición de* τ,
- *existe unha variable* x, *tal que* $\sigma/\pi = x = \tau/\pi'$ *(respectivamente* $\sigma/\pi' = x$ *se* π' *é de* σ*),*
- *existen posicións* $\omega, \omega', \upsilon$, $\omega \parallel \omega'$, *tal que* $\pi = \omega \cdot \upsilon, \pi' = \omega' \cdot \upsilon$,
- *existen posicións* $\pi_0, \ldots, \pi_n$ *tal que* $\pi = \pi_0$, $\pi_0 \parallel \pi_1 \parallel \ldots \parallel \pi_n = \pi'$.

A seguinte propiedade mostra unha forma de detectar que para a obtención dunha solución poido ser utilizada unha derivación alternativa.

4.17 *Sexa* $\Gamma = \{\sigma \stackrel{?}{=} \tau\}$ *un problema de E-unificación, que se ten resolto mediante a aplicación de* $\stackrel{na}{\Rightarrow}$ *na posición* π *de* σ, *coa 2-célula* α. *Se se verifican as seguintes condicións :*

- *as condicións* $\delta \stackrel{?}{=} \theta$ *son resoltas por* ρ,
- λ *é o unificador máis xeral de* $\sigma\rho[\pi \leftarrow t(\alpha)\rho] \stackrel{?}{=} \tau\rho$
- τ/π *non é variable (ou está conectado cunha expresión non variable diferente de* $t(\alpha)$*).*

entón existe unha secuencia alternativa de Narrowing con Abstracción que proporciona $\rho\lambda$ *usando* α^{-1}.

Proba. A aplicación de α en σ/π da orixe a $<\sigma[\pi \leftarrow t(\alpha)], \delta> \stackrel{?}{=} <\tau, \theta>$. Aténdese á posición π na que foi aplicada α. Se τ/π non é variable, existe $\mu = mgu(\tau\rho/\pi, \sigma[\pi \leftarrow t(\alpha)]\rho)$, de onde, pola separación das variables (*3.6*), necesariamente τ/π e $t(\alpha)$ son unificables, polo que se pode aplicar α^{-1} en τ/π obtendo $<\sigma, \delta_1> \stackrel{?}{=} <\tau[\pi \leftarrow s(\alpha)], \theta_1>$ onde $\delta_1 = \Big(mgu((\tau/\pi)', t(\alpha)\Big)\theta_1$, $\theta_1 = abst(\tau/\pi)$.

Dado que os operadores comúns en $\sigma \stackrel{?}{=} \tau[\pi \leftarrow s(\alpha)]$ coinciden, pode aplicarse $\stackrel{d}{\Rightarrow}$ ata que sexan eliminados estes (realmente só é necesario para os que se atopan en posicións π', $\pi' \leq \pi$ ou $\pi \leq \pi'$). Esto produce $<\sigma, \delta_1> \stackrel{?}{=} <\tau[\pi \leftarrow s(\alpha)], \theta_1> \stackrel{d}{\Rightarrow} <\sigma_1, \delta_1, \sigma_2> \stackrel{?}{=} <\tau_1, \delta_1, \tau_2>$. Cada variable $x_\alpha \in cod(s(\alpha))$ que se atope en máis dunha ecuación de τ_2, é eliminada ata xenerar as ecuacións correspondentes a $\delta \stackrel{?}{=} \theta$. Xa que ρ se obtén ó resolver estas ecuacións, aplícase este en $<\sigma_1, \delta_1> \stackrel{?}{=} <\tau_1, \theta_1>$ ata obter o propio λ, xa que por hipótese os operadores eliminados en $\sigma \stackrel{?}{=} \tau[\pi \leftarrow s(\alpha)]$ son eliminados tamén na obtención de λ, dado que $\lambda = mgu(\sigma\rho[\pi \leftarrow t(\alpha)\rho] \stackrel{?}{=} \tau\rho)$.

No caso de que τ/π sexa variable, aplícase igualmente α^{-1} en tódalas posicións non variables conectadas. O razonamento para este caso é semellante. □.

4.3.7 Comparación con outros métodos

Os métodos *top-down* teñen un sistema de actuación moi diferente ós que seguen o esquema *down-up* e as derivacións obtidas non se atopan incluídas unhas nas outras, en forma análoga ó que sucede con *outermost narrowing* e *innermost narrowing*. Sen embargo poden observarse algunhas ventaxas de empregar métodos *down-up* :

- nestes, a semellanza dun termo co lado da regra a aplicar é moito maior que no caso *top-down*. Esto nótase especialmente cando existen varias regras co mesmo operador superior.

  Exemplo 4.23
  $R = \{f(x, g(x, b)) \rightarrow h(x),\ f(x, g(x, c)) \rightarrow h(x),\ f(x, g(x, d)) \rightarrow h(x)\}$,
  $\Gamma = \{f(x, g(x, d)) \stackrel{?}{=} h(x)\}$.

- A proximidade destes procedementos á *narrowing*, e polo tanto (lema de levantamento) á propia derivación que mostra a igualdade das instanciacións, permite incorporar ós métodos *down-up* estratexias de *narrowing* para eliminar novas redundancias, como se ten mostrado para o *basic narrowing*.

Respecto dos procedementos *down-up* mellórase o propio *Relaxed Narrowing* tanto no senso de impoñer condicións máis fortes para a súa aplicación de condicións, como por permitir ecuacións condicionais. En relación co método de Moser[129] obsérvase que as condición son en parte máis débiles que o propio $\mathcal{T}_{\mathcal{CP}}$, se ben son incorporadas novas restriccións, algunhas das cales foron incorporadas a este novo método.

Comparado co procedemento de Ohsuga e Sakai, detéctase unha millora notoria en canto se evita explicitar a complección dos pares críticos, polo que se encamiña directamente o proceso á unificación do problema. Por outra parte mostróuse como se poden incorporar algunhas das estratexias de narrowing o novo procedemento, polo que as posibles ventaxas que o emprego directo de narrowing son aproveitadas.

Os resultados experimentais, que aparecen no Apéndice B, mostran unha considerábel reducción en tempo e espacio en comparación co *Relaxed Narrowing* o cal é, por outra parte, unha mellora directa do procedemento de Unificación Ecuacional de Gallier e Snyder $\mathcal{T}_{\mathcal{LP}}$.

Capítulo 5

Conclusións e Líneas de Investigación

5.1 Conclusións

Mostróuse un novo instrumento de unificación ecuacional, para tratar con teorías para as que o *narrowing* non é apropiado por fallar a completitude.

Para a obtención do novo algoritmo, empregóuse a técnica da *abstracción* para adaptar o método de Dougherty e Johann *Relaxed Narrowing* á unificación de pares de substitucións. Na unificación destes pares, o control das variables é máis estricto que no habitual esquema de pares de termos, e probóuse que a abstracción de variables das ecuacións non son necesarias.

Esta mellora, interprétase tamén como unha forma de correxir a simetría que *Relaxed Paramodulation* establece entre o lado esquerdo da regra ecuacional e o subtermo do problema dado onde se aplica tal regra. Xa que, se ben a unificación é unha operación conmutativa, o tratamento a dar ás expresións antes mencionadas, non debe selo, xa que os subtermos do problema dado poden ser previamente modificados pero en cambio esto non se pode facer cos subtermos da regra ecuacional. Esto mismo pode decirse do *basic narrowing* respecto de *narrowing*, dado que no primeiro prohíbense as aplicacións de ecuacións en posicións que previamente teñen correspondido a posicións de variables de ecuacións usadas previamente, mentras que o segundo non fai distinción entre as instancias das variables de ambas expresións, lado esquerdo dunha ecuacións da teoría e termo do problema dado.

Por outra parte, entre tódolos métodos de E-unificación *down-up* completos en teorías xerais, o que aquí se desenrola é o máis próximo ó narrowing, ó que permite adaptar estrategias completas deste, ó novo algoritmo. Pode observarse igualmente, que imita o uso de pares críticos, case como se estes tiveran sido xerados, sen precisar a complección destes pares. Esto ten a vantaxe de que este proceso de complección non é, en xeral, finitario (e pode acabar con fallo) amais de que polo proceso de complección poden xerarse un gran número de pares críticos sen relevancia para un problema de

E-unificación dado.

O enfoque categórico, permite observar certos detalles (e.g. o papel das variables) que noutro contexto cecais pasasen desapercibidos. Permite observar que algunhas propiedades empregadas na unificación son casos particulares de operacións xa coñecidas e ás que se poden adaptar resultados doutros contextos, posto que a que a teoría de categorías permite unha integración de lingoaxes diferentes.

A unificación ecuacional, directa o indirectamente, tén sido empregada noutros campos de investigación como a proba automática e a programación lóxica con igualdade. En ambos casos, necesítanse mecanismos para tratar coa igualdade de forma que o número de solucións redundantes sexa reducido no maior grado posible. De feito, algúns métodos de unificación ecuacional xurden de aproximacións ó tratamento da igualdade no campo da proba automática e noutros casos sucede xustamente o proceso inverso como na *paramodulation* con axiomas de igualdad[38] e o método de E-unificación de Moser [129] para o primeiro caso e o Cálculo RPC de Snyder e Lynch[172] para o segundo.

5.2 Líneas de investigación

Consideramos que deste traballo conéctase cunha serie de problemas de interese :

- Realizar unha experimentación incorporando as melloras da sección 4.3.6 có fin de dar unha medida destas. Con estas melloras, conseguirase reducir aínda máis o número de solucións se ben a cambio dun maior custe en tempo e espacio. Esta incorporación, obriga a unha modificación importante das estructuras de datos empregadas respecto dos programas utilizados no Apéndice A, especialmente para a incorporación da estrategia tipo *basic narrowing*. Interesa amais, empregar representacións máis concisas que aceleren o proceso.

- Investigar acerca de novos refinamentos que poidan ser engadidos ó *Narrowing con Abstracción*. Especialmente en canto a incorporar novas estratexias de conmutación e orientación. Amais ambos problemas van, en certa medida parellos: trátase de partir dunha derivación cunha solución dada, e alterar o orden no que se aplican as transformacións para eliminar solucións repetidas. Unha das maiores dificultades consiste en atopar unha medida que permita probar a completitude : ó cambiar o orden dunha derivación pode producirse un incremento do número ou tamaño das novas ecuacións ou ben do número de regras a aplicar. Un exemplo da dificultade deste tipo de problemas, é o da *Eliminación Ambiciosa de Variables* (*Eager Variable Elimination*). Se λ é un E-unificador da expresión $\Gamma \cup \{x \stackrel{?}{=} t\}, x \notin Var(t)$, dada a sustitución $\rho = \{x/t\}$ naturalmente se verifica $\rho \leq_E \lambda_{\{x\}}$, pode pensarse que toda solución do problema será obtida de $\Gamma\rho$ sen necesidade de usar ecuacións en t. Sen embargo é coñecido que se λ tén sido obtido logo de utilizar algunhas ecuacións en t antes de eliminar x, nalgúns

casos esto da lugar a un menor número de aplicacións de novas ecuacións en Γ e, en moitos casos tamén a un un menor número de variables despois do uso destas ecuacións. Gallier e Snyder [72] detectaron un erro na proba de Hölldobler onde usaba a completitude de esta estrategia. Elo é unha mostra da dificultade de modificar o orden utilizado nunha derivación dada e un campo aberto para a mellora dos algoritmos de unificación ecuacional.

- Como con outros mecanismos de unificación ecuacional, deberá investigarse a forma de incorporar este novo método a outros campos onde se faga preciso tratar coa igualdade en presencia de teorías non canónicas. Na proba automática, tense empregado a estrategia do *conxunto soporte* [186] para impedir a xeración de consecuencias dadas exclusivamente a partir das cláusulas que pertenecen ó conxunto soporte. Se estas cláusulas son ecuacións, esta estrategia non permite a producción de pares críticos a partir de esas ecuacións. O mecanismo de resolución empregado, debe imitar o uso de esas consecuencias que non son xeneradas, en forma semellante ó uso de pares críticos en unificación ecuacional.

- Xa que este traballo trata acerca dunha nova aproximación ó tratamento da igualdade en condicións o máis xerais posibles, pode conectarse con todos aqueles campos nos que se precise o emprego de axiomas sen restriccións previas a éstos, ou ben con condicións bastante débiles.

Bibliografía

[1] Aho A.V., Hopcroft J.E., Ullman J.D. The design and analysis of computer algorithms. Addison Wesley Pub. Comp. , 1974.

[2] Albert L., Casas R., Fages F., Zimmermann P. Average Case analysis of unification algorithms. Raports de Recherche N. 1213 INRIA, 1990.

[3] Alpuente M., Falaschi M., and Manzo F. Analyses of Inconsistency for Incremental Equational Logic Programming. Proceedings PLILP'92. LNCS 631, pp.443-457, 1992.

[4] Alpuente M., Falaschi M.,Ramis M.J., and Vidal G. Narrowing aproximations as an Optimization for Equational Logic Programs. Proceedings PLILP'93. 1993.

[5] Asperti A., Longo G. Categories, Types, and Structures MIT, 1991.

[6] Baader F. Unification in idempotent semigroups is of type zero. *Journal of Automated Reasoning 2*, pp.283-286, 1986.

[7] Baader F. Characterizations of unification type zero. Proceedings RTA'89, LNCS 355. pp. 2-14, 89.

[8] Baader, F. Unification in commutative theories. *J. of Symbolic Computation 8*, pp.479-498, 1989.

[9] Baader, F. Unification, weak unification, upper bound, lower bound and generalization problems, SEKI report SR-90-02, Univ. Kaiserlautern,1990.

[10] Baader F. Unification in Varieties of Completely Regular Semigroups. Proceedings IWWERT'90, LNCS 572. pp.210-230,1992.

[11] Baader F. Schulz K.U. Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. Proceedings CADE'92, LNAI 607, pp.50-65, 1992.

[12] Baader F., Nutt W. Adding Homomorphism to Commutative/Monoidal Theories. Proceedings RTA'91, LNCS 488, pp.124-135, 1990.

[13] Bachmair L., Dershowitz N. and Hsiang J. Orderings for Equational Proofs. *Proc. First Symp. Logic in Computer Science IEEE,* pp.346-357, 1986.

[14] Bachmair L., Ganzinger H., Lynch C., Snyder W. Basic Paramodulation and Superposition. Proceedings CADE'92, LNAI 607, pp.462-476, 1992.

[15] Barr M., Wells C. Category Theory for Computing Science. Prentice Hall 1990.

[16] Baxter L. An efficient unification algorithm. Technical report CS-73-23, Univ. Waterloo, 1973.

[17] Bidoit M., Corbin J. A Rehabilitation of Robinson's Unification Algorithm. *Information Processing 83, North Holland,* pp.909-914, 1983.

[18] Birkhoff G. On the Structure of Abstract Algebras. *Proc. Cambridge Phil. Soc* 31, pp.433-454, 1935.

[19] Bläsius K.H., Siekman J.H. Partial Unification for Graph Based Equational Reasoning. Proceedings CADE'88, LNCS 310 , pp. 396-414, 1988.

[20] Bockmayr A. A Note on a Canonical Theory with Undecidable Unification and Matching Problem. *Journal Automated Reasoning 3.* pp.379-381, 1987.

[21] Bockmayr A. Conditional Narrowing Modulo a Set of Equations. *Applicable Algebra in ECC* 4, pp.147-168, 1993.

[22] Bockmayr A., Krischer S., Werner A. An Optimal Narrowing Strategy for General Canonical Systems. Proceedings CTRS'92 LNCS 656, pp.483-497, 1993.

[23] Book R., Siekman J. On unification : Equational theories are not bounded. *Journal Symbolic Computation 2.* pp.317-324, 1986

[24] Bosco P.G., Giovannetti E., Moiso C. Refined Strategies for Semantic Unification. Proceedings TAPSOFT'87 LNCS 250, pp.276-290, 1987.

[25] A. Boudet. Unification in a Combination of Equational Theories: an Efficient Algorithm. Proceedings CADE'90, LNCS 449, pp.292-307, 1990.

[26] Boudet A. Contejean E. On n-Syntactic Equational Theories. Proceedings AlP'92, LNCS 632, pp.446-457, 1992.

[27] Boudet, A. Contejean E., Devie H. A new AC unification algorithm with a new algorithm for solving diophantique equations. Proceedings 5th IEEE Symposium on Logic in Computer Science, pp. 289-299, 1990.

[28] Boudet A., Jouannaud J.P., Schimdt-Schauß M. Unification in boolean rings and abelian groups. *J. of Symbolic Computation 8,* pp.449-477, 1989.

[29] Boyer R.S., Moore J.S. The sharing of structure in theorem-proving programs. *Machine Intelligence* 7, 1972.

[30] Buchberger B. History and Basic Features of the Critical-Pair/Completion Procedure. *J. of Symbolic Computation 3*, pp.3-38, 1987.

[31] Bürckert H.J. Lazy E-unification- a method to delay alternative solutions. Technical report, Univ. Nancy, 1987.

[32] Bürckert H.J., Herold A., Scmidt-Schaub M. On Equational Theories, Unification and (Un)Decidability. *Journal of Symbolic Computation 8*, pp.3-50, 1989.

[33] Burris S., Sankappanavar H.P. A Course in Universal Algebra, Springer-Verlag, 1981.

[34] Cachafeiro Chamosa L.C. Un Algoritmo Paralelo de Unificación. Technical Report n. Dep. Computación Univ. A Coruña, 1994.

[35] Cachafeiro Chamosa L.C., Freire Nistal J.L. Abstracción de variables y Narrowing en Unificación Ecuacional . Technical Report n. 6 Dep. Computación Univ. A Coruña, 1992.

[36] Cachafeiro Chamosa L.C., Freire Nistal J.L. Implementación de un Nuevo Método General de Unificación Ecuacional. Proceedings PRODE'93, pp. 111-122, 1993.

[37] Chabin J., Réty P. Narrowing directed by a graph of terms. Proceedings RTA'91, LNCS 488, pp.112-123, 1991.

[38] Chang C., Lee R.E. Symbolic Logic and Mechanical Theorem Proving Academic Press 1973.

[39] Chang S. Towards a Theorem Proving Architecture. M.S. Thesis Inst. Technology Pasadena, 1981.

[40] Citrin W. Parallel Unification Scheduling in Prolog. Ph.D. Dissertation EECS Dep. Univ. California, 1988.

[41] Clausen M. Fortenbacher A. Efficient Solution of Linear Diophantine Equations. *J. of Symbolic Computation* 8. pp.201-216, 1989.

[42] Colmerauer A. Prolog and Infinite Trees. Logic Programming Academic Press, pp.231-251, 1982.

[43] Colomb R.M. Enhancing Unification in Prolog through Clause Indexing. *J. of Logic Programing 10*, pp.23-44, 1991.

[44] Comon H. Unification et disunification. Theories et applications. *J. of Symbolic Computation 7*, pp.371-425, 1989.

[45] Comon H., Lescanne P. Equational Problemas and Unification *J. of Symbolic Computation 7*, pp.371-425, 1989.

[46] Cook S. A. An Overview of Computational Complexity. *Com. Ass. Comp. Mach. 26*, pp.400-408, 1983.

[47] Courcelle B. Fundamental properties of infinite trees. *Theoretical Comput. Sci. 25*, pp.95-169, 1983.

[48] Davis M. The prehistory and early history of automated deduction. *Automation Reasoning 1, Classical Papers on Computational Logic 1957-1966.* Springer-Verlag, 1983.

[49] De Champeaux D. About the Paterson-Wegman Linear Unification Algorithm . *Journal Computer and System Sciences 32*, pp.79-90, 1986.

[50] Delsart B. A New Approach to General E-Unification Based on Conditional Rewriting Systems. Proceedings CTRS'92. LNCS 656, pp.468-482, 1992.

[51] Dershowitz N. Termination of Rewriting. *J. of Symbolic Computation 3*, pp.69-116, 1987.

[52] Dershowitz N., Jouannaud J.P. Rewrite Systems. Handbook of Theoretical Computer Science North-Holland, pp.245-320, 1992.

[53] Dershowitz N., Sivakumar G. Solving goals in equational languages. Proceedings 1th CTRS. LNCS 308, pp.45-55, 1987.

[54] Diaconescu R. Contraction Algebras and Unification of (Infinite) Terms. *J. of Comput. Syst. Sci. 44*, pp.23-42, 1992.

[55] Dincbas M., van Hentenryck P. Extended unification algorithms for the integration of functional programming into logic programming. *Journal Logic Programming 4(3)*, pp.199-219, 1987.

[56] Dougherty D., Johann P. An improved general E_unification method. Proceedings CADE'90, LNCS 449, pp.261-275, 1990.

[57] Dwork C., Kanellakis P., Mitchell J. On the sequential nature of unification. Technical report CS-83-26 Brown University, 1983

[58] Echahed R. Uniform Narrowing Strategies. Proceedings ALP'92, LNCS 632 pp.259-275, 1993.

[59] Eder E. Propieties of Substitutions and Unification. *J. of Symbolic Computation 1*, pp.31-46, 1985

[60] Escalada-Imaz G., Ghallab M. A practically efficient and almost linear unification algorithm. *Artificial Intelligence 36*, pp.249-263, 1988.

[61] Fages F. Associative-conmutative unification. Proceedings CADE'84, LNCS 170. pp. 194-208, 1984.

[62] Fages F., Huet G. Complete Sets of Unifiers and Matchers in Equational Theories. Proceedings CAAP'83, LNCS 159. pp. 205-220, 1983

[63] Fages F., Huet G. Complete Sets of Unifiers and Matchers en Equational Theories. *Theoretical Computer Science 43(1)*, pp.189-200, 1986.

[64] Fay M. First Order Unification in an Equational Theory. Procedings CADE'79, LNCS 87, pp.161-167, 1979.

[65] Fernandez R., Freire J.L. Teorías Algebraicas Morita Equivalentes. Universidade de A Coruña, 1991.

[66] Fortenbacher A. An Algebraic Approach to Unification under Associativity and Conmutativity. Proceedings RTA'85, LNCS 202 pp. 381-397, 1985.

[67] Freire Nistal J.L. Elementos de Programación Funcional y Categorías. Ed. Vía Láctea. A Coruña 1990.

[68] Freire Nistal J.L., Alonso Amo F., Aguado Martín F. On the Naturality of the Iteration and Recursive Specifications. *Applied Mathematics and Computation* 57, pp.1-18, 1993.

[69] Freire Nistal J.L., Aguado Martín F. Naturality of the Conditional and Recursion. *Intern. Journal Computer Mathematics 35*, pp.7-14, 1990.

[70] Fribourg L. Slog : a logic programming language interpreter based on clausal superposition and rewriting. *Proceedings 2th Int. Symp. on Logic Programming, 4*, pp.172-185. 1985.

[71] Gallier J.H., Snyder W. A general complete E-unification procedure. Proceedings RTA'87. LNCS 256, pp.216-227, 1987.

[72] Gallier J.H., Snyder W. Complete Sets of Transformations for General E-unification. *Theoretical Computer Science 67*, pp.202-260, 1989.

[73] Gallier J.H., Snyder W. Higher-Order Unification Revisited : Complete Sets of Transfomations *J. of Symbolic Computation 8*, pp.101-140, 1989.

[74] Giovannetti E., Levi G., Moiso C., Palamidessi C. Kernel-Leaf A Logic plus Functional Language *Journal of Computing Systems Sciences 42*, pp.139-185, 1991.

[75] E. Giovannetti, C. Moiso. A completness result for E-unification algorithms based on conditional narrowing. Proceedings FLFP, LNCS 306, pp. 157-167, 1987.

[76] Goguen J., Meseguer J. EQLOG : Equality, types, and generic modules for logic programming. *Functional and Logic Programming* , pp. 295-363, 1986.

[77] Goldschalager L.M. The monotone and planar circuit value problems are log space complete for P. In *SIGACT News 9*, pp.25-29, 1977.

[78] Gollakota R. Design and Analysis of UNIFIC: A Coprocessor for the Unification Algorithm. M.S. Thesis Texas A&M Univ. 1986.

[79] Guard J.R. Automated Logic for Semi-Automated Mathematics. Scientific Report Air Force Cambridge Lab AD602 710. 1964.

[80] Herbrand J. Investigations in Proof Theory. En *Logical Writtings*. Harvard Univ. Press, pp.44-202, 1971.

[81] Herold A. Some Basic Notions of First Order Unification Theory. Univ. Karlsruhe Internal Report, 1983.

[82] Herold A. Combination of unification algorithms. Proceedings CADE'86, LNCS 230, pp. 450-469, 1986.

[83] Hölldobler S. Foundations of Equational Logic Programming. LNAI 353, 1989.

[84] Hsiang J., Kirchner H., Lescanne P., Rusinowitch M. The Term Rewriting Approach to Automated Theorem Proving. *J. of Logic Programming 14*, pp.71-99, 1992.

[85] Huet G. Resolution d'equations dans des langages d'ordre 1,2..ω (en francés). Thèse d'Etat Univ. Paris VII, 1976.

[86] Huet G. An algorithm to generate the basis of solutions to homogeneus linear Diophantine equations. *Inf. Process Lett.* 7(3), pp.144-147, 1978

[87] Huet G. Abstract Properties and Applications to Term Rewriting Systems *Journal of Ass. Comp. Mach. 27(4),* pp.797-821, 1980.

[88] Huet G. A complete proof of correctness of the Knuth-Bendix completion algorithm. *J. of Comput System Sci. 23*, pp. 11-21, 1981.

[89] Huet G. Formal Structures for Computation and Deduction. Course notes. Carnegie-Mellon Univ. 1986.

[90] Huet G., Oppen D.C. Equations and Rewrite Rules : A Survey. Formal Languages :Perspectives and Open Problems. Academic Press, 1980

[91] Hullot J.H. Canonical Forms and Unification. Proceedings CADE'80, LNCS 87, pp.318-334, 1980.

[92] Jaffar J. Efficient unification over infinite terms. *New Generation Computing 2*, pp.207-219, 1984.

[93] Jaffar J., Lassez J.L., Maher M.J. A theory of complete logic programs with equality *Journal Logic Programming* 3. pp.211-223, 1984.

[94] Jouannaud J.P. Kirchner H. Completion of a set of rules modulo a set of equations, *SIAM J. Comput. 15*, pp.1155-1194, 1986.

[95] Jouannaud J.P., Kirchner C. Solving equations in Abstract Algebras: A Rule-Based Survey of Unification. *Computational Logic: Essays in Honor of A. Robinson*, 1991.

[96] Jouannaud J.P., Muñoz M. Termination of a Set of Rules Modulo a Set of Equations. Proceedings CADE'84. LNCS 170, pp.175-193, 1984.

[97] Kanellakis P.C., Revesz P.Z. On the Relationship of Congruence Closure and Unification. *J. of Symbolic Computation* 7, pp.427-444, 1989.

[98] Kaplan S. Fair conditional term rewriting systems : Unification, termination and confluence. Technical report 194, Univ. Orsay, 1984.

[99] Kapur D., Krishnamoorthy M.S., Narendran P. A New Linear Algorithm for Unification. Internal report 82CRD-100, General Electric, 1982.

[100] Kapur D., Narendran P. Matching, Unification and Complexity. *SIGSAM Bull* 21,4 pp.6-9, 1987.

[101] Kapur D., Sivakumar G., Zhang H. A new method for proving Termination of AC rewrite systems. Proceedings Foundations of Software Technology and Theor. Comp. Sci., LNCS 472, pp.133-148, 1991.

[102] Kelly G.M., Street R. Review of the Elements of 2-Categories. Proc Category Seminar, LNM 420, 1974.

[103] Kirchner C. Méthods et Outils de Conception Systématique d'Algoritmes d'Unification dans les Théories equationelles (en francés). Thèse d'Etat, Univ. Nancy 1985.

[104] Kirchner C. Kirchner H. Implementation of a general completion procedure parameterized by built-in theories and strategies. Proceedings the EUROCAL, 1985.

[105] Kirchner C., Klay F. Syntactic theories and unification *5th IEEE Symp. on Logic in Computer Science*, pp. 270-277, 1990.

[106] Klay F. Undecidable properties of syntactic theories. Proceedings RTA'91 LNCS 488 , pp.136-149, 1991.

[107] Knight K. Unification : A multidisciplinary survey. *Ass. for Comp. Mach. Surveys* 21, pp.93-124, 1989. March 1989.

[108] Knuth D.E., Bendix P.B. Simple Word problems in universal algebras. Computational Problemas in Abstract Algebra. Pergamon Press, pp.263-297, 1970.

[109] Krischer S., Bockmayr A. Detecting Redundant Narrowing Derivations by the LSE-SL Reducibility Test. Proceedings RTA'91, LNCS 488 pp.74-85, 1991.

[110] Lankford D.S. Canonical Inference. Technical Report, South-western University, 1975.

[111] Lankford D. Non-negative Integer Basis Algorithms for Linear Equations with Integer Coeficients. *Journal of Automated Reasoning* 5, pp.25-35, 1989.

[112] Lankford D.S., Ballantine A.M. Decision procedures for simple equational theories with permutative axiomas : Complete sets of permutative reductions. Technical report Univ. Texas, 1977.

[113] Lassez J.L., Maher M., Marriot K. Unification revisited. Proceedings FLFP, LNCS 306, pp.67-114, 1987.

[114] Lincoln P., Christian J. Adventures in Associative-Conmutative Unification *J. of Symbolic Computation 8* pp.217-240, 1989

[115] Livesey M Siekmann J. Unification of sets and multisets. Technical report 3/76 Institut für Informatik I, Universität Karlsruhe, 1976.

[116] Mac Lane S. Categories for the Working Mathematician. Springer-Verlag, 1971.

[117] Manes E.G. Algebraic Theories, G.T.M. 26, Springer-Verlag, 1976.

[118] Markov A.A: The impossibility of some algorithms in the theory of associative systems (en ruso). *Dokl. Akad. Nauk SSSR 55*, pp.587-590, 1947.

[119] Martelli A., Montanari U. An efficient unification algorithm. *Ass. Comp. Mach. Transaction on Programming Languages and Systems 4(2)*, pp. 258-282, 1982.

[120] Matijasevič Y. Diophantine representation of recursively enumerable predicates. *Actes Congrès International des Mathématiciens 1* pp.235-238, 1970

[121] Mauny M. Functional programming in CAML, Technical report, INRIA, 1991.

[122] McDermott J. A Rule Based Configurer of Computer Systems. En *Artificial Intelligence 19*, pp.39-88, 1982.

[123] Meseguer J. Rewriting as a unified model of concurrency. Technical Report SRI-CSL-90-02R, SRI International, 1990.

[124] Meseguer J., Goguen J., Smolka G. Order-sorted Unification. Repor CSLI-87-86, 1987.

[125] Meseguer J., Winkler R. Parallel Programming in Maude. Research Directions in High-level Parallel Programming Languages. LNCS 574, pp.253-293, 1992.

[126] Middeldorp A., Hamoen E. Counterexamples to Completeness Results for Basic Narrowing. Proceedings ALP'92, LNCS 632, pp.244-258,1992.

[127] Mitchell B. Rings with several objects. *Advances in Mathematics, 8*, pp.1-161, 1972.

[128] Moreno-Navarro J.J., Artalejo M.. Logic Programming with Functions and Predicates : The Language BABEL. *Journal Logic Programming 12,* pp. 191-223, 1992.

[129] Moser M. Improving Transformation Systems for General E-Unification. Proceedings RTA'93. LNCS 690, pp. 92-105, 1993.

[130] Mukai K. A Unification Algorithm for Infinite Trees. Procedings IJCAI-83, pp.547-549, 1983.

[131] Narendran P., Otto F. Some results on equational unification. Proceedings CADE'90, LNCS 449. pp.276-291, 1990.

[132] Newman M.H.A. On theories with a combinatorial definition of "equivalence" *Ann. Math.* 43, pp.223-243, 1942.

[133] Nipkow T. Unification in primal algebras : their powers and their varieties *Journal of Ass. Comp. Mach. 37*, pp.742-76, 1990.

[134] Nutt W. The Unification Hierarchy is Undecidable. SEKI-Report SR-89-06, Universität Kaiserlautern 1989

[135] Nutt W., Réty P., Smolka G. Basic narrowing revisited. *J. of Symbolic Computation 7*, pp.295-318, 1989.

[136] Ohsuga A., Sakai K. Complete Equational Unification Based on an Extension of the Knuth-Bendix Procedure. LNCS 572, pp. 197-209, 1992.

[137] Paterson M.S., Wegman M.N. Linear Unification. *Journal of Computer and Systems Sciences 16*, pp.158-167, 1978.

[138] Peterson G., Stickel M. Complete Sets of Reductions for Equational Theories with Complete Unification Algorithms. *Journal of Ass. Comp. Mach. 28-2.* pp.233-264, 1981.

[139] Plotkin G. Building-in Equational Theories. En *Machine Intelligence 7*, pp. 73-90, 1972.

[140] Post E.L. Recursive unsolvability of a problem of Thue. *J. of Symbolic Logic 13*, pp.1-11, 1947.

[141] Pottier L. Minimal solutions of linerar diophantine systems: bounds and algorithms. Proceedings RTA'91, LNCS 488, pp.162,173, 1991.

[142] Power A.J. An abstract formulation for rewrite systems. Proceedings Seminar on Category Theory and Computer Science, LNCS 389, pp.300-312, 1989.

[143] Prawitz D. An Improved Proof Procedure. *Theoria* 26, 1960.

[144] Project Formel. The CAML reference manual. INRIA, 1989.

[145] Ramamohanarao K., Shepeherd J. A Superimposed Codeword Indexing Scheme for Very large Prolog Data Bases. 3th Int. Conference on Logic Programming. Imperial College of Science and Technology, 1986.

[146] Reddy U. Narrowing as the operational semantics of functional languages. *Proceedings the IEEE Symposium on Logic Programming* , pp.138-151, 1985.

[147] Réty P. Improving Basic Narrowing Techniques. Internal Report 87-R-012 Centre de Recherche en Informatique de Nancy 1987.

[148] Réty P., Kirchner C., Kirchner H., Lescanne P. NARROWER : A new Algorithm for Unification and its Applications to Logic Programming. Proceedings RTA'85, LNCS 202. pp.141-157, 1985.

[149] Ringeissen C. Unification in a combination of equational theories with shared constants and its application to Primal Algebras. Proceedings LPAR'92, LNCS 624, pp.261-272, 1992.

[150] Robinson J.A A Machine-Oriented Logic Based on the Resolution Principle. *Journal of Ass. Comp. Mach.* 12, pp. 32-41, 1965.

[151] Robinson J.A. Computational logic: The unification computation *Machine Intelligence 6*, pp.63-72, 1971.

[152] Robinson J.A. Logic : Form and Function - The Mechanization of Deductive Reasoning. Nort-Holland, 1979.

[153] Robinson G.A., Wos L. Paramodulation and Theorem Proving in First Order Theories with Equality. *Machine Intelligence 4*, pp.135-150, 1969.

[154] Robinson P. The Sum : An AI Coprocessor. *Byte 10*, pp.169-180, 1985.

[155] Rosen B.K. Tree-manipulating systems and Church-Rosser theorems. *Journal Ass. Comp. Mach. 20*, pp.160-187, 1973.

[156] Rusinowitch M. Path of Subterms Ordering and Recursive Decomposition Ordering Revisited. *J. of Symbolic Computation 3*, pp.117-131, 1987.

[157] Rusinowitch M. Vigneron L. Associative Commutative Deduction. Abstracts of Construction of Computational Logics CCL'92 pp. 41, 1992.

[158] Ruzicka P., Privara I. An Almost Linear Robinson Unification Algorithm. *Acta Infomatica 27*, pp.61-71, 1989.

[159] Rydeheard D. Funtors and Natural Transformations. Proceedings the Workshop on CTCP'86, LNCS 240, pp.43-50, 1986.

[160] Rydeheard D., Burstall R. A categorical unification algorithm Proceedings the Workshop on CTCP'86, LNCS 240, pp.493-505, 1986.

[161] Rydeheard D., Burstall R. *Computational Category Theory Prentice Hall*, 1988.

[162] Rydeheard D., Stell J. A categorical Treatment of Equational Proofs and Unification Algorithms. Proceedings Category Theory and Computer Science, LNCS 283. Springer Verlag, pp. 114-139, 1987.

[163] Schmidt-Schauß M. Unification in a combination of arbitrary disjoint equational theories. Proceedings CADE'88, LNCS 310, pp.378-396, 1988.

[164] Schmidt-Schauß M. Unification in a Combination of Arbitrary Disjoint Equational Theories. *J. of Symbolic Computation 8*, pp.51-99, 1989.

[165] Schmidt-Schauß M. Unification in Permutative Equational Theories is Undecidible. *J. of Symbolic Computation* 8, pp.415,421, 1989.

[166] Sibai F. N. Parallel Unification : Theory and Implementations Proceedings of Parallelization in Inference Systems, LNAI 590, pp.51-80, 1991.

[167] Siekmann J. Unification of commutative terms. Conference on Symbolic and Algebraic Comp. LNCS 72. pp.531-545, 1976

[168] Siekman J. Unification Theory : a Survey. *J. of Symbolic Computation 7*, pp. 207-274, 1989.

[169] Siekman J.H., Szabó P. Universal Unification. Proceedings CADE'94, LNCS 170, pp.1-42, 1984.

[170] Slagle J.R. Automated Theorem Proving for Theories with Simplifiers, Commutativity and Associativity. *Journal of Ass. Comp. Mach. 21,* pp. 622-642, 1974.

[171] Smolka G., Ait-Kaci H. Inheritance Hierarchies :Semantics and Unification. *J. of Symbolic Computation 7*, pp.343-370, 1989.

[172] Snyder W., Lynch C. Goal Strategies for Paramodulation. Proceedings RTA'91, LNCS 488. Springer Verlag, pp.150-161, 1991.

[173] Steinbach J. Improving associative path orderings. Proceedings CADE'90. LNAI 449, pp.411-425, 1990.

[174] Stell J.G. Modelling Term Rewriting Systems by Sesqui-Categories Technical Report TR94-02 University of Keele, 1994.

[175] Stickel M. E. Unification Algorithms for Artificial Intelligence Languages. Ph.D thesis, Carnegie-Mellon University, 1976.

[176] Stickel M.E. A Unification Algorithm for Associative-Conmutative Functions. *Journal of the Ass. Comp. Mach. 28*, pp. 423-434, 1981.

[177] Szabo P. The undecidability of the DA-unification problem. Technical report Univ. Karlsruhe, 1979.

[178] Szabo P. Theory of First Order Unification (en alemán). Ph. D. Thesis, Univ. Karlsruhe, 1982.

[179] Tiden E. Unification in combinations of collapse-free theories with disjoints sets of function symbols. LNCS 230, pp.431-449, 1986.

[180] Venturini-Zilli M. Complexity of the unification algorithm for first-order expressions. Research report Consiglio Nazionale delle Recherche Roma, 1975.

[181] Vitter J.S., Simons R.A. New Classes for Parallel Complexity. A Study of Unification and Other Complete Problemas for P. IEEE *Transaction on Computers* pp. 403-418, 1986.

[182] Warren D. H. An Abstract Prolog Instruction Set. SRI Technical Note 309, 1983.

[183] Wikström A. Functional Programming Using Standard ML. Prentice Hall. Cambridge. 1987.

[184] Williams J.G. Instantiation Theory. LNAI 518, 1991.

[185] Wise M.W., Powers D.H. Indexing Prolog Clauses via Superimposed Code Words and Field Encoded Words. *Proceedings Int. Sym. on Logic Programming IEEE*, pp.203-210, 1984.

[186] Wos L., Mc Cune W. Automated theorem proving and logic programming : A natural symbiosis. *Journal Logic Programming* 11, pp.1-53, 1991.

[187] Yamamoto A. Completness of Extended Unification Based on Basic Narrowing. Proceedings LP'88, LNAI 383, pp.1-10, 1989.

[188] Yelick K. Combining unification algorithms for confined equational theories. Proceedings RTA'85, LNCS 202, pp. 365-380, 1985.

[189] Yelick K. Unification in Combination of Collapse-Free Regular Theories. *J. of Symbolic Computation*, 3:153-181, 1987.

[190] You J.H. Enumerating Outer Narrowing Derivations for Constructor-Based Term Rewriting Systems. *J. of Symbolic Computation* 7, pp.319-341, 1989.

Apéndice A

Implementación do procedemento de E-unificación

Para realizar a comparación entre *Relaxed Narrowing* e *narrowing con abstracción*, teñen sido implementados ambos na lingoaxe de programación funcional CAML de INRIA, version 3.1.

Para elo, téñense empregado algunhas definicións de tipos e funcións que aparecen no traballo de Rydeheard e Burstall[161].

Esta implementación, cuio código móstrase a continuación, atópase dividido en cinco partes :

- Funcións preliminares e Analizador,
- Unificación sintáctica de substitucións,
- Unificación sintáctica en casos particulares,
- Funcións auxiliares para a E-unificación,
- Funcións que proporcionan E-unificadores.

Na primeira delas, convírtense as cadeas de caracteres iniciais nos obxectos da sintaxe abstracta (substitucións e pares de substitucións). Para elo, utilízase unha adaptación das funcións *build* de Wikström [183], xunto cunha adaptación da transformación de expresións a termos que aparece en el trabajo de Huet [89]. Recíprocamente, dase unha forma para pasar as solucións resultantes á lingoaxe concreta inicial. Nesta lingoaxe as ecuacións son escritas na forma seguinte :

$$\{termo_1 = termo'_1, \ldots, termo_n = termo'_n\}$$

onde *termo* vén dado por

- $operador(argumento_1, \ldots, argumento_m)$ para un operador de aridade m,

- as variables exprésanse por símbolos que se inician por unha das letras u, v, x, y, z,
- as constantes non necesitan o emprego de parénteses.

Pode observarse esta sintaxe no exemplo da páxina 213.

Na lingoaxe abstracta, as variables son obxectos do tipo `int`, para simplificar o tratamento de variables (e.g. para o coproducto) en forma semellante á que se utiliza no mencionado traballo de Huet [89].

A segunda parte, non difiere esencialmente da que aparece en [161], adaptándoo á propia lingoaxe CAML e correxindo un erro na función : `irreducible_unify`. Sen esta corrección, existen algunhas situacións de non unificabilidade que non eran detectadas.

Na terceira parte constrúense as funcións que realizan a unificación de substitucións en casos particulares : `unif_nrepeat_vars` realiza a unificación sintáctica no caso de que o problema sexa lineal (non existen variables repetidas). Este é xustamente o caso da unificación do par $(\sigma/\pi)', (s(\alpha))'$ que sucede na transformación $\stackrel{rp}{\Rightarrow}$. Na transformación $\stackrel{na}{\Rightarrow}$, non se ten unha situación tan forte, e só a linealidade nunha das expresións $(\sigma/\pi)'$. Para dar unha forma semellante á anterior (xa que por atoparse separadas non se transmiten as instancias), emprégase `unif_fact`, que factoriza o unificador, como se precisa en 4.10.

Na parte seguinte desenrólanse as funcións necesarias para a E-unificación aquí exposta e.g. a función `abs_occ` que realiza a abstracción.

Por último móstrase o código das funcións que corresponden ós algoritmos de E-unificación comparados : *Narrowing con Abstracción* e *Relaxed Narrowing con Abstracción*, tanto a versión que inclúe a operación *Descomposición*, como aquela que nona emprega. Esto se realiza respectivamente mediante :

- `rel_abs_narrow` para *Relaxed Narrowing con Abstracción*,
- `narrow_abs` para *Narrowing con Abstracción*,
- `rel_abs_narrow_dec` para *Relaxed Narrowing con Abstracción* e uso de *Descomposición*. Esta corresponde a con *Relaxed Narrowing*,
- `narrow_abs_dec` que realiza a E-unificación mediante *Narrowing con Abstracción* e a *Descomposición*.

Existe unha versión deste programa no cal as funcións se atopan incluídas nun fichero *.EXE do Sistema Operativo DOS, o que proporciona os E-unificadores dun problema dado, sen precisar que ese ordenador estea disponible o propio CAML-LIGHT.

A relación de `rel_abs_narrow` e de `narrow_abs_dec` coas principais funcións construídas, pode observarse na figura A.1 para a función `narrow_abs` e na figura A.2 para `narrow_abs_dec`.

Figura A.1: Funcións e chamadas recursivas en narrow_abs

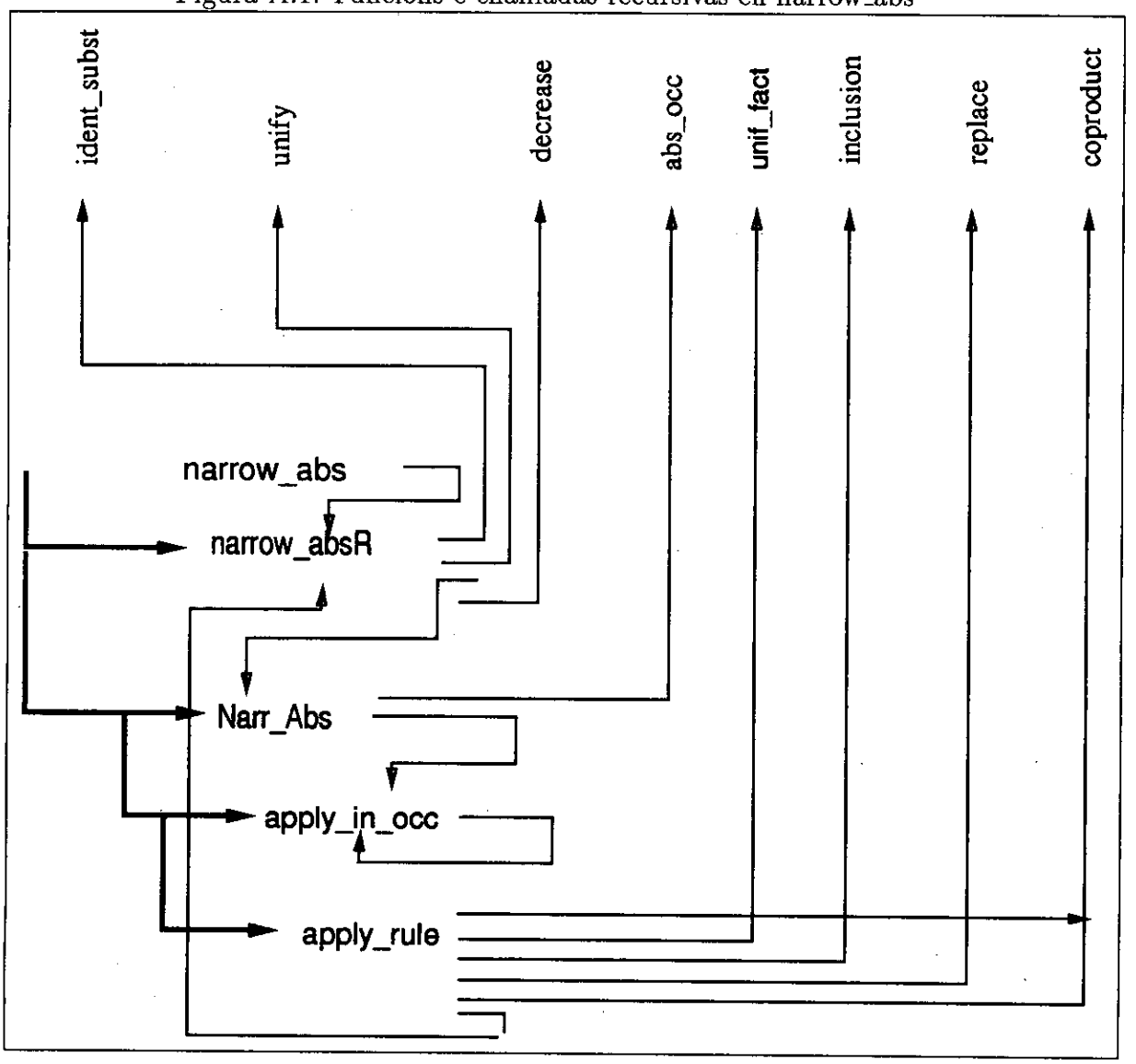
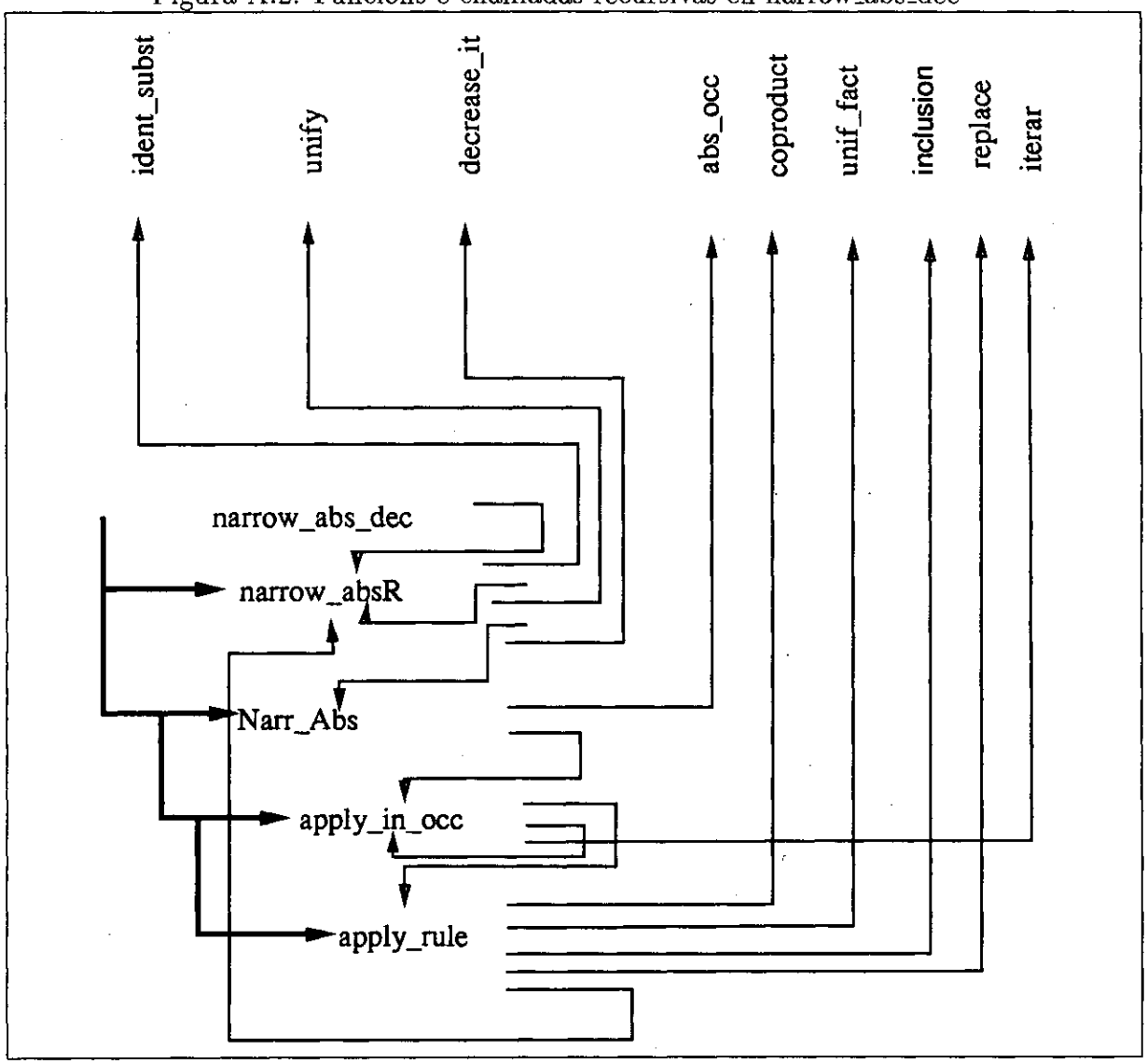


Figura A.2: Funcións e chamadas recursivas en narrow_abs_dec



Algoritmo A.1 *Entrada : O par de substitucións a unificar ecuacionalmente e a presentación Saida : Unha lista coas solucións obtidas (se remata)*

Narrow_Abs(R, σ, τ) =
  Se $\sigma = \tau$ *Enton* $\{1_X\}$
    Se existe $\rho = mgu(\sigma, \tau)$ *Enton* $\{\rho\} \cup$ *Narr_Abs*(σ, τ)
    Noutro caso Narr_Abs(σ, τ)

onde
Narr_Abs(σ, τ) =
  Sexa $F = \{\sigma, \tau\}$, $F' = F$ *e* $S = \emptyset$
  Mentras $F' \neq \emptyset$ *seleccionar* $\eta \in F'$ *(e sexa* $\mu \in F \backslash \{\eta\}$*)*
    Sexa $\Pi =$ { *posicions non variables de* η}
    Mentras $\Pi \neq \emptyset$ *seleccionar unha posicion* π *de* Π.
    Sexa $R' = R \cup R^{-1}$
      Mentras $R' \neq \emptyset$ *seleccionar un elemento* α *de R' /* realizando o coproducto das suas variables con* $Var(\sigma, \tau)$*) */*
      Sexa $\theta =$ *substitucion abstraccion de* η/π
        Se existe $\rho = mgu\Big((\eta/\pi)', s(\alpha)\Big)$ *Enton*
          Sexa $S = S\Big(\cup\, j_X$
          Narrow_Abs$(R, < \eta \circ j_X[\pi \leftarrow t(\alpha) \circ j_{X_\alpha}], \rho\theta >, < \mu, \theta >) \Big)$
        Noutro caso $S = S$
      $R' = R' - \{\alpha\}$
    $F' = F' - \{\eta\}$
  $\Pi = \Pi - \{\pi\}$
S

```
(* ---- eunifi.ml  Luis Carlos Cachafeiro Chamosa ----- *)

(* ------  UNIFICACION ECUACIONAL EN CAML          ------ *)

(* ------- TIPOS  E FUNCIONS PREVIAS E
                    ANALIZADOR (PARSER) ---------------- *)

(* Esta parte ten definidas as funcions :

union --------  une duas listas sen repeticions
seteq --------  chequea a igualdade de listas
```

```
max   --------  calcula o valor maximo dun par
subst_apply ---  aplica unha sustitucion a un termo
comp  ---------  compon duas sustitucions
ident ---------  realiza a sustitucion identidade
inclusion -----  produce a sustitucion que inclue un conxunto
                 en outro
i_term --------  devolve o valor i-esimo dunha lista
disj_list -----  realiza a disxuncion dos valores
(booleans)       dunha lista
flatlist  -----  aplasta unha lista de listas de mesmo tipo
max_list ------  calcula o valor maximo dunha lista
minus ---------  elimina un elemento dunha lista
from_to -------  construe a lista que vai de un no outro
coproduct -----  separa os elementos de duas listas e inclueos

                 nunha nova (coas correspondentes
                 inclusions e funcion universal)
explode -------  separa unha cadea en caracteres
build_digit ---  construe un termo variable dada unha serie
de               caracteres
build_term ----  construe un termo sexa variable ou non
build_op ------  construe os termos non variables
build_term2 ---  construe o termo dado o primeiro caracter do

                 seu operador
build_term3 ---  construe o termo dado o primeiro operador
build_eq ------  devolve a primeira ecuacion dun sistema
build_eq' -----  devolve a primeira ecuacion (iniciada ou non)
build_system' -  devolve un sistema dende a primeira ecuacion
build_system -   construe un sistema de ecuacions dada unha
                 lista de caracteres
build  --------  construe un sistema dada unha cadea de
                 caracteres
escribe  ------  imprime por pantalla un simbolo dado
display_term --  saca por pantalla un termo
display_termset  mostra na pantalla un conxunto de termos
                 dados
display_Subst -  saca por pantalla unha sustitucion
output_Sol ----  saca por pantalla unha lista de sustitucions

                 ------------------------------------ *)
```

```
exception lista_nula;;
exception erro_sintaxe;;
exception non_componibles;;

let  rec union = fun ([],b) -> b
 |((x::a),b) ->[x]@(union((subtract  a [x]),(subtract b [x])))
;;
let rec seteq =
 fun ([],b)->b=[]
 |((x::a),b)->seteq(subtract a [x],subtract b [x]);;

let max x y  = if x>y then x else y;;

type element == int ;;
type opr = Opr of string*(element list);;
type term = Var of element|Apply of opr*(element ->term);;
type substitution = Subst of (element list)*(element ->term)
*(element list);;

let rec subst_apply =
 fun (Var x)->  (fun (Subst(_,f,_))-> f x)
 |(Apply(psi,s)) -> fun S ->Apply(psi,fun x->
                                    subst_apply (s x) S);;

let rec comp ((Subst(c,g,d)) as S,Subst(a,f,b)) =
  if seteq(b,c)  then Subst(a,(fun x ->subst_apply (f x) S),d)

  else raise non_componibles;;

let ident a = Subst(a,(fun x->Var (x)),a);;
let inclusion  (c,b) = Subst(c,(fun x -> Var x), b);;

let rec i_term = fun [] -> raise lista_nula
                    |(x::xs)-> (fun 1->x
                                    |i->i_term xs(i-1));;

let rec disj_list l = list_it (fun e l -> e or l) l false;;

let   flatlist  = flat_map (fun x ->x);;

let max_list = it_list max 0  ;;
```

```
let minus (a,x) = subtract a [x] ;;

let rec from_to (m,n) = if m>n then [] else
m::from_to(m+1,n);;

(* dado un par de conxuntos de enteiros, extrae o coproducto
(union disxunta) de eles *)

let coproduct (a,b) =
 if a=[] then
  let   j_a  =  Subst([],(fun _ ->raise lista_nula) ,b)
  in let  j_b = ident b
  in let univ (f,g) = g
 in ((b,(j_a,j_b)),univ)
 else let  x = max_list (a)
      in let cop_ab = union(a,(map(fun y ->x+y)b))
      in let j_a = Subst(a,(fun y -> Var y),cop_ab)
      in let j_b = Subst(b,(fun y -> Var (y+x)),cop_ab)
      in let  univ (f,g) =
          let  (Subst(_,funf,c)) = f
         in let (Subst(_,fung,_)) = g
         in let   fun_copr  y = if y > x then (fung(y-x))
                                else (funf y)
      in Subst(cop_ab,fun_copr,c)
in ((cop_ab ,(j_a,j_b)),univ)   ;;

(* explode convirte unha cadea nunha lista de simbolos *)

let rec explode =
 fun "" ->[]
  |st ->
    let x = nth_char st 0
    and st'= sub_string st 1 (-1+string_length st)
  in if (int_of_char x) = 32 then explode st'
     else (char_for_read x)::(explode st')
 ;;

(* build_digit convirte unha expresion nun termo variable
e actualiza o "diccionario" anterior :
Fd transforma secuencias de simbolos en elementos (variables)
Df   "   variables nos simbolos correspondientes
n = ultimo numero (elemento) empregado *)
```

```
let rec build_digit (Fd,Df,n) =
 (fun x ->
  (fun []-> raise erro_sintaxe
  |(x1::xs)->
   if  x1 =","  or x1 = ")" or x1 = "}" or x1 = "=" then
    try (let m = Fd x in ((Fd,Df,n),Var (m),x1::xs)
        ) with _ -> let Fd' z = if z =x then n else Fd z
          in let Df' m = if m = n then x else Df m
          in ((Fd',Df',n+1),Var(n),x1::xs)
   else build_digit (Fd,Df,n) (x^x1) xs
  )
 ) ;;

(* A partir de  D(iccionario) previo e da correspondiente
serie de simbolos construe :
(novo diccionario, termo, resto de simbolos)
utiliza build_op que construe termo iniciado por caracter
        build_term2 que construe termo dado o simbolo
        build_term3 que reune os argumentos (subtermos)
                    dun termo para construir este
*)

let rec build_term  D =
 fun [] ->raise erro_sintaxe
 |(x::xs) ->
 let  asciii = int_of_char (nth_char x 0 )
in if  asciii>116 & asciii <= 122 then build_digit  D x xs
   else build_op D x xs

and build_op D x =
 (fun  [] -> raise erro_sintaxe
  |(x1::xs) ->
    if x1 = "=" or  x1 = "," or x1 = ")" or x1 = "}"
    then (D,(Apply(Opr(x,[]),(fun _ -> raise lista_nula)))

         ,(x1::xs))
    else  if x1 = "(" then build_term2  D (x,xs)
          else build_op  D (x^x1) xs
 )

and build_term2  D (opp,xs) =
```

```
 match xs with [] -> raise erro_sintaxe
 |(x2::xs2) ->
   if x2 = ")" then (D, Apply(Opr(opp,[]),
                        (fun _ -> raise lista_nula)) ,xs2)
   else let (D',lr,cs) = build_term D xs
 in build_term3  D' (opp, [lr] ,cs )

and build_term3 D (x ,llr ,cs ) =
 let  (y::css) = cs
in if y = ")" then  (D, Apply(Opr(x,from_to(1, (list_length

                llr))),i_term llr),css)
   else let  (D', ti,ccss)  = build_term D css
in build_term3 D' (x,(llr@[ti]),ccss)       ;;

(* build_eq dado o  D(iccionario) e lista de simbolos
construye (Novo D(iccionario), (termo,termo),resto lista)
utiliza build_eq' tal que :
se a lista e vacia (no hai un primeiro termo da ecuacion)
noutro caso construir ecuacion acaba construindo o
   seguinte termo
*)

let rec build_eq D  ec =    build_eq' D [] ec

and
build_eq' D =
 (fun [] ->
   (fun [] ->raise erro_sintaxe
    |(x::ecs)->
      let  (D',r,cs) =  build_term  D (x::ecs)
    in build_eq'  D' [r] cs
   )
  |ac ->
    (fun [] ->raise erro_sintaxe
     |(x::ecs ) ->
       let   (acc::_ ) = ac
      in let (D', t,cs)  =    build_term  D ecs
      in  (D',((acc,t),cs))
    )
 )     ;;
```

```
(* build_system' dado un Diccionario previo, unha lista coas
ecuaciones (sustituciones)  procesadas e unha lista cos
simbolos sen procesar, incorpora as sustituciones a
seguinte ecuacion (amais de actualizar o Diccionario (so
queda con Df que permite "recuperar" a notacion inicial) *)

let rec
build_system' D  acc =
   (fun [] -> raise erro_sintaxe
    |(x::ss) ->
       if x = "}"  then let  (_,Df,_) = D
        in  (Df,acc )
        else
         if x = "," then
          let (D',(r,cs)) = build_eq D  ss
          and (Subst(a,f,b),Subst(_,g,_)) = acc
          in let (_,_,n) = D'
          and (r1,r2) = r
          and max = list_length a
          in let p_subst_ac =
           (Subst(a@((max +1)::[]),
              (fun x -> if x <= max then f x else r1),
              from_to(1,n-1)) ,
            Subst(a@((max +1)::[]),
                (fun x -> if x <= max then g x  else r2),
              from_to(1,n-1)))
          in build_system'  D' p_subst_ac cs
         else raise erro_sintaxe
  ) ;;

(* buil_system e a funcion que construe finalmente o par
de sustitucions e o diccionario, a partir da lista de
simbolos inicial *)

let rec build_system  =
 (fun [] -> raise lista_nula
  |(e1::ss ) ->
   let  (((_,_,n)as D),(r,cs)) = build_eq ((fun _ -> raise
lista_nula),
                              (fun _ -> raise lista_nula),1) ss

   in let (r1,r2) = r
```

```
  in build_system'  D (Subst([1],(fun _ ->
r1),from_to(1,n-1)),
                    Subst([1],(fun _ -> r2),set_ec))
from_to(1,n-1))   ;;

let  build car  = build_system ( explode car);;

let escribe x = print_char (char_of_int(x+48));;

(* A partir do Diccionario (funcion que adxudica simbolos a
elementos) e un termo, mostra este *)

let display_term D s  =
 let rec display_term_string =
  fun (Var x) -> ( try D(x) with  _ ->
"x"^(char_for_read(char_of_int x ))
                  )
  |(Apply(Opr(ro,s),f))  ->
    ro^"("^(display_termset  (s, f))^")"

and
display_termset  (s, f )=
 if s=[] then ""
 else let  e::s' = s
in if (s'=[]) then display_term_string  (f e)
   else (display_term_string  (f e ))^" , "^(display_termset

      (s',f))
in print_string (display_term_string s);;

(* mostra a sustitucion dada a partir do diccionario *)

let rec  display_Subst D (Subst(a,f,b)) =
 if a=[] then print_string ""
 else let  (i::a') = a
in if a'=[] then (print_string"   ";display_term D (Var i)
                  ;print_string" -> "; display_term D (f i))
   else (print_string"   ";display_term D (Var i) ;
         print_string" -> ";
         display_term  D (f i) ; print_string " ,  "
         ;print_newline(); display_Subst D (Subst(a',f,b))   )
 ;;
```

```
(* Dado o diccionario D e a lista coas solucions,
mostra cada unha das sustitucions *)

let  output_Sol D  Sol =
 let rec output_Sol' =
  fun [] _ ->  ()
  |(x::[]) n ->
  let  saida =
   if n < 10 then  (print_int(n);print_string "  SOLUCION ";
                    print_newline();
                    display_Subst  D x)
   else (print_int(n / 10); print_int(n mod 10);
         print_string "  SOLUCION  " ;print_newline();
         display_Subst  D x)
  in  saida
  |(x::Sol) n ->
   let  saida =
    if n < 10 then  (print_int(n);print_string"  SOLUCION ";
                     print_newline();
                     display_Subst  D x)
    else (print_int(n / 10); print_int(n mod 10);
          print_string"  SOLUCION  "; print_newline();
           display_Subst D x)
  in  (saida ;print_newline();output_Sol' Sol(n+1))
in print_string "[";( output_Sol' Sol 1) ; print_string "]";
print_newline() ;;


(* ------------ FIN INTRODUCCION E ANALIZADOR ------------- *)

(* ------------ UNIFICACION BURSTALL E RYDEHEARD ---------- *)
(* moi pocas diferencias coas funcions de Computational
Category Theory *)
(* Esta parte ten definidas as funcions :

restrict --- restrinxe o dominio dun par de sustitucions
occurs ----- chequea a presencia dunha variable nun
               termo
app_Subst -- aplica unha sustitucion a un termo
arity_op --- devolve a aridade dun operador dado
pp_source -- indica o dominio dun par de sustitucions
```

```
pp_target -- o rango dun par de sustitucions
top_same --- chequea se dous termos tenhen o mesmo operador
               superior
witness ---- chequea se duas sustitucions tenhen os mesmos
               operadores no primeiro par de termos
sum_coequal - emprega un procedemento de descomposicion e
               calculo nos factores e composicion del
               resultado
sum_descomp -- descompon un par de sustitucions en dous pares
               de elas
composite_coequalize
          ---  esquece a parte epica na extraccion do
               coigualador dun par acompanhado dunha parte
               epica comun
composite_decompose
          ---  extrae unha sustitucion epica inicial (non iso)
               dun par de sustitucions
unit_unify --- obten o unificador dun par variable-
               termo
irreducible_unify
          ----  obten o unificador dun par sinxelo de
               sustitucions
unify -------- realiza a unificacion de sustitucions

                  ------------------------------------  *)

(* A funcion restrict dun conxunto e un par de
sustitucions extrae a nova sustitucion cuio dominio  o
conxunto dado *)

let restrict (b,P) =
 let  (f,g) = P
in let (Subst(_,f',c)) = f
in let (Subst(_,g',d)) = g
in (Subst(b,f',c),Subst(b,g',d))   ;;

(* chequea se un termo conten unha variable dada *)

let rec occurs =
fun  (z ,(Var y)) ->  z = y
| (z,(Apply(phi,f) as t)) ->
  let  (Opr(_,s)) = phi
```

```
  in let  terms = map f s
in  not( s=[]  ) &
disj_list (map (fun x -> occurs(z, f x)) (s))   ;;

exception  occ_not_exist;;
exception non_def;;
exception ciclo;;
exception conflicto_operadores;;
exception non_unif;;
exception erro_sintactico;;
exception sum_descompos;;
exception testigo    ;;
exception no_unificables ;;

(* app_Subst aplica unha sustitucion a un termo *)

let  app_Subst f x =
 let  (Subst(_,f',_)) = f
in f' x   ;;

(* aridade do operador *)

let arity_op (Opr(_,s)) = s   ;;

let pp_source P =
 let (Subst(a,_,_)) = fst P in a;;

let pp_target P =
 let (Subst(_,_,b)) = fst P in b;;

(* chequea se o operador superior de ambos termos e o
mesmo *)

let top_same (t1, t2 ) =
 match (t1,t2) with (( Apply(phi,_)),(Apply(pho,_))) ->
  let  (Opr(ri,s)) = phi
  in let (Opr(ro,t)) = pho
 in ri = ro & (list_length s = list_length t)
|(_,_) -> false ;;

(* chequea se o primeiro par de termos de duas sustitucions
tenhen o mesmo operador *)
```

```
let witness ((f,g)as P) =
 let  a =  pp_source P
in if a=[] then raise lista_nula
   else let  (x::_) =  a
in if (top_same ((app_Subst f x) ,(app_Subst g x)))
   then x else raise testigo      ;;

(* dados dous pares de sustituciones aplica cq (sera unify en
xeneral) o primeiro par e a sustitucion resultante
componna co segundo par e volta a aplicar cq. O valor e
a composicion de ambos resultados *)

let sum_coequal  cq ((f,g),(f',g')) =
 let  q = cq(f,g)
 in let r = cq (comp(q,f'),comp(q,g'))
 in let s_coeq = comp(r,q)
in s_coeq   ;;

(* rompe un par de sustitucions nun par simple (dous
termos) e o resto. Falla se as sustitucions non tenhen
dominio polo menos 2 *)

let sum_descomp P =
 let  a = pp_source P
in if 2> list_length(a)  then raise lista_nula
   else let (b::c) =  a
in (restrict([b],P),restrict(c,P))     ;;

(* Cando h e epica, o coigualador da composicion de h
con cada sustitucion de P e o propio coigualador de P *)

let composite_coequalize (cq)(P,h) = cq(P);;

(* composite_decompose dun par de sustitucions, extrae unha
sustitucion epica comun a ambas (se existe) e o par de
sustitucions restante *)

let composite_decompose ((f,g) as P) =
 let  x = witness (P)
 in let (Apply(phi,s))  = app_Subst (f) (x)
 in let (Apply(pho,t)) = app_Subst (g) (x)
```

```
 in let c = arity_op(phi)
 in let ff = (fun  y  -> s( y ))
 in let gg = (fun y ->  t(y ))
in ((Subst(c,ff,pp_target P),Subst(c,gg,pp_target P)),
    Subst(pp_source P,(fun _ -> Apply(phi,(fun z  ->  Var
z))),c))   ;;

(* crea a sustitucion que leva a variable x no termo
t,  de dominio b e rango c *)

let unit_unify (x,t,b,c) =
 let  un_unif = Subst(b,(fun z -> if z = x then t else Var
z),c)
in un_unif  ;;

(* se P es directamente unificable (dous termos un de eles
variable), extrae o correpondente unificador. Ten unha
pequena diferencia con de Bur &Ryd xa que no de eles a
excepcion if not .. non se realiza, o que provoca que en
certos casos unify ten exito cuando non debera,
concretamente se o  primeiro par e unificable pero non o resto

*)

let irreducible_unify ((f,g)as P) =
 let  b = pp_target P
in if (pp_source P= []) then ident b
   else let  (x::a') = pp_source P
   in if not(a' =[]) then raise no_unificables
      else let  s = app_Subst (f) (x)
           and t =app_Subst(g)(x)
      in
       match (s,t) with
          (Var(z),Var(z')) ->
           let  c =  if z = z' then  b
                     else subtract b [(max z z') ]
          in unit_unify((max z z'),Var ((min z z')),b,c)
         |(Var(z),t') ->
             if occurs (z,t') then raise ciclo
              else unit_unify(z,t',b,minus(b,z))
         |(s',Var(z)) ->
            if occurs (z,s') then raise ciclo
```

```
            else unit_unify(z,s',b,minus(b,z))
        | (_,_) -> raise conflicto_operadores       ;;

(* Esta funcion extrae o unificador dun par de
sustitucions dada *)

let rec  unify P =
 try (let  (P',P'') = sum_descomp P
in  sum_coequal  (unify)(P',P'')   )
with _ -> try (let  (P',h) = composite_decompose (P)
in composite_coequalize  (unify)(P',h) )
with _ ->  irreducible_unify P ;;

(* ----------- FIN ALGORITMO DE UNIFICACION
                  RYDEHEARD E BURSTALL      --------------- *)

(* -----------  OUTRAS FORMAS DE UNIFICACION ---------------
*)

(* Esta parte produce as funcions unif_nrepeat_vars e
unif_fact empregadas en unificacions particulares. *)


(* Emprega as seguintes funcions :

unif_nrepeat_vars - realiza a unificacion en pares sen
                    variables repetidas
particular -------- extrae o termo mais especifico entre un
                    anterior e outro actual
partic_set -------- acumula os termos mais especificos
                    dunha serie de argumentos dun termo dado
xeneral_part ------ calcula o termo mais xeneral e mais
                    especifico a partir dun par dado e un
                    novo termo
xeneral_part_set -- obten o t. mais xeneral e mais especifico
                    dun anterior e os novos argumentos
unif_fact' -------- realiza a unificacion factorizada para un
                    par de termos
unif_fact --------- realiza a unificacion factorizada dun
                    par de sustitucions

                    ------------------------------------  *)
```

```
(* unif_nrepeat_vars realiza a unificacion de duas
sustitucions coa propiedade de non ter variables
repetidas. Polo tanto non necesita nen chequear a existencia
de ciclos nen realizar a instanciacion das variables
eliminadas *)

let rec unif_nrepeat_vars (f,g) =
 let  (Subst(a,funf,b)) = f
 and (Subst(_,fung,_)) = g
in if a=[] then ident b
  else let  (x::a') =   a
  in if not(a'=[]) then
       let  (Subst(_,funu,b')) = unif_nrepeat_vars (
            Subst([x], funf, b), Subst([x], fung, b))
        in let (Subst(_,funu',b'')) =
               unif_nrepeat_vars(Subst(a',funf,b'),
                                 Subst(a',fung,b'))
     in Subst(b,(fun x -> if not(mem x b') then funu x
           else if mem x b'' then (Var x) else  funu' x),b'')

     else let  s = funf x
          and  t = fung x
     in match s with (Var m) ->
         Subst(b,(fun x -> if x = m then t else
                           Var x),subtract b [m])
     | (Apply(Opr(ro,sro),funro)) ->
       match t with (Var m) ->
        Subst(b,(fun x -> if x = m then s else
                          Var x),subtract b [m])
     |(Apply(Opr(ro',_  ),funro')) ->
       if ro <> ro' then raise conflicto_operadores else
     unif_nrepeat_vars(Subst(sro,funro,b),Subst(sro,funro',b))

  ;;

(* particular de f1 (funcion que indica a posible
instanciacion e conxunto de variables non eliminadas), o valor
anterior mais especifico (particular) e o novo, intenta a
extraccion de posibles novas condicions (a incorporar a f1)
xunto co  novo termo mais especifico *)
```

```
let rec particular f1 sp s =
 match (s,sp) with (Var x,Var y)  ->
  let  f1' v = if v = x then sp else fst(f1) v
  and b = subtract (snd f1) [x]
 in ((f1',b),sp)
 |(Apply(Opr(ro,sro),funro),Var y) ->
   let  f1' v = if v = y then s else fst(f1) v
   and b = subtract (snd f1) [y]
  in ((f1',b),s)
 |(Var y,Apply(_,_)) ->
   let  f1' v = if v = y then sp else fst(f1) v
   and b = subtract (snd f1) [y]
  in ((f1',b),sp)
 |(Apply(Opr(ro,sro),funro),Apply(Opr(rop,_),funrop)) ->
   if ro<>rop then raise conflicto_operadores
   else let (f1',funp) =
    partic_set(funrop,funro) (f1, fun _ ->raise non_def) sro
  in (f1', Apply(Opr(ro,sro),funp))

and partic_set (funsp,funss) =
 let rec partic_set' (f1,funp) sro =
     if  sro=[] then (f1,funp)
     else let  (i::sro') =   sro
          in let (f1',sp') = particular f1 (funsp i)(funss i)
   in partic_set' (f1',fun j ->if j = i then sp' else funp j)
sro'
in partic_set'   ;;

(* general_part de :
     - f1 (instanciaciones e variables non eliminadas previas,

     - un par (instanciacion mais xeneral, instanciacion mais
          especifica) de termos unificados por algunha
          variable (da regla)
     - o novo termo a unificar con esa variable,

obten a nova instanciacion e conxunto de variables,
o novo  par (termo mais xeral, termo mais especifico)
o ter que facer iguais a variable da regla (que non
aparece aqui), o par antiguo, e o novo  termo. *)

let rec general_part f1 (sx,sp) s =
```

```
 match (sx,s) with (Var x,Var y) ->
  let  f1' z = if z = y then sx else fst(f1) z
  and b = subtract (snd f1) [y]
 in ((f1',b),sx,sp)
 |(Var x, Apply(Opr(ro,sro),funro)) ->
   let  (f1',sp') = particular f1 sp s
 in (f1',sx,sp')
 |(Apply(Opr(rox,_),funrox),Var x) ->
   let  f1' y = if y = x then sx else fst(f1) y
   and b = subtract (snd f1) [x]
  in ((f1',b),s,sp)
  |( Apply(Opr(rox,_),funrox) , Apply(Opr(ro,sro),funro)) ->
   if ro<>rox then raise conflicto_operadores
   else let  (Apply(_,funrop)) = sp
   in let (f1',funx,funp) = general_part_set
                          (funrox,funrop,funro)
        (f1,(fun _ ->raise non_def),(fun _ ->raise

                          non_def))sro
      in (f1',Apply(Opr(ro,sro),funx),Apply(Opr(ro,sro),funp))


and general_part_set (funrox,funrop,funro) =
 let rec general_part_set' (f1,funx,funp) sro =
   if  sro=[] then (f1,funx,funp)
   else let  (i::sro') =   sro
        in let (f1',sx,sp) = general_part f1
                          (funrox i,funrop i) (funro i)
   in general_part_set'
      (f1',(fun j ->if j = i then sx else funx j),
           (fun j ->if j=i  then sp else funp j)) sro'
in general_part_set'   ;;

(* unif_fact e unha funcion iterativa que realiza a
unificacion factorizada de dous termos (o primeiro sen
variables repetidas e o segundo pode telas) tendo como
as correspondientes instanciaciones a f1 e f2.

     Aqui f1 leva conta das instanciacions das
     variables dos termos da esquerda (problema) e
     as variables non eliminadas.
```

```
     f2 leva conta das instanciacions realizadas sobre
     as variables dos termos da dereita (reglas),
     mediante o control do seu  termo mais xeneral e
     especifico *)

let rec unif_fact' (s,t) (f1,f2) =
 match (s ,t) with (Var x,_) ->
  let  f1' y = if x =y then t else fst(f1) y in let b =
subtract (snd f1) [x]
 in ((f1',b),f2)
 |(Apply(Opr(ro,sro),funro),Var(x_rule)) ->
  ( try ( let  (sx,sp) = f2 x_rule
    in let (f1',sx',sp') = general_part f1 (sx,sp) s
    in let f2' y = if y = x_rule then (sx',sp') else f2 y
in let b = subtract (snd f1) [x_rule]
   in ((fst f1',b),f2')
      )
   with non_def ->
    let  f2' y = if y = x_rule then (s,s) else f2 y
in let b = subtract (snd f1) [x_rule]
   in ((fst f1,b),f2')    )
 |(Apply(Opr(ro,sro),funro) ,Apply(Opr(rot,srot),funrot)) ->
      if ro <> rot then raise conflicto_operadores
    else list_it (fun i -> unif_fact' (funro i,funrot i))sro
(f1,f2)  ;;

(* unif_fact realiza a unificacion factorizada dun par de
sustituciones dadas, supondo implicitamente que a da
esquerda non ten variables repetidas. Chama a unif_fact'
para realizar aa unificacion de cada problema "simple" (pares
de termos) *)

let unif_fact (f,g) =
 let  (Subst(a,funf ,b)) = f
 in let (Subst(_,fung,_)) = g
 in let (f1,f2) =   list_it(fun i -> unif_fact' (funf i,fung
i)) a
        (((fun _ ->raise non_def),b),(fun _ ->raise non_def))
in Subst(b,(fun x -> try (fst(f1) x) with _ -> try(fst(f2 x))
with _ -> Var x),
         snd(f1))
     ;;
```

```
(* ------------ FIN UNIFICACION SINTACTICA ---------------- *)

(* -------- FUNCIONS PRELIMINARES E-UNIFICACION ---------- *)

(* Definense unha serie de funcions que van a ser utilizadas
na unificacion ecuacional *)

(* As funcions definidas son :

replace ------ modifica unha sustitucion incorporando un
               termo nunha posicion dada
Subst_null --- e a identidade na lista nula
decrease ----- aplica en forma decrecente un procedemento
               que produce listas sobre as que debe compoer
               o resultado e aplicar de novo o
               procedemento
abs_occ ------ realiza a abstraccion posicion dun termo
abs_occ_set -- realiza a abstraccion a partir dos
               argumentos dun termo
build_axioms - construe unha lista de 2-celulas dada unha
               cadea de caracteres
equal_term --- chequea a identidade de dous termos
equal_term_set chequea a identidade dos argumentos
ident_Subst -- chequea a identidade de duas sustitucions

               ---------------------------------------- *)

(* replace reemplaza nunha sustitucion, unha posicion dada,
por un subtermo (presente como sustitucion sinxela). Se
aparecen novas variables, actualiza o rango da
sustitucion resultante *)

let rec replace f occ g =
 let  (Subst(a,funf,b)) = f
 and (Subst(c,fung,d)) = g
in let (x::a') =  a
 and(y::c') =   c
in match occ with [] -> f
|(i::oc) ->
 if i <> 0 then
```

```
  let  fi = Subst([1],( fun 1 -> funf i| _ ->raise non_def),

                b)
 in let (Subst(_,funfi,_)) =  replace fi oc g
 in Subst(a,(fun j -> if j <> i then funf j else (funfi i)),

       union(b,d) )
 else
  match funf x with (Var _) ->
   if oc = [] then Subst(a,(fun x1 -> fung(y+x1-x)),

              union(b,d))
   else raise occ_not_exist
  |Apply(Opr(ro,sro),funro) ->
   if oc =[] then Subst(a,(fun x1 -> fung(y+x1-x)),

                         union(b,d))
   else
    let  funi (i::occ1) j =
      if j<> i then funro j
       else let rec  funi' =
             fun [] ->(fun  _ -> fung y)
             |(i'::occ2)->
              (fun (Var _) -> raise occ_not_exist
               |(Apply(Opr(roi,sroi),funroi)) ->
                Apply(Opr(roi,sroi),
                      (fun j -> if j<> i' then funroi j else
                                funi'  occ2 (funroi i'  ))  )
              )
   in funi'  occ1 (funro i)
in Subst([x],(fun z ->if z=x then Apply(Opr(ro,sro),(funi oc))
                      else raise non_def ) ,
        union(b,d))     ;;

let Subst_null = Subst([],(fun _ -> raise lista_nula),[]);;

(* decrease e unha especie de "expansion" do metodo empleado
para a funcion sum_coequal : se naquil caso cq producia un
determinado valor, e aplicabase o resto do problema, aqui
univ_unifR produce unha lista da que cada elemento
componese co resto para aplicar nuevamente a funcion o
novo problema *)
```

```
let decrease (univ_unifR) (f,g)  =
 let  (Subst(a,_,_)) = f
in if a=[] then Subst_null::[]
   else let   (x::a') =   a
        in let  Sol = univ_unifR (restrict([x],(f,g)))
        in let (f'',g'') = restrict(a', (f,g))
in flatlist(
    map(fun sigma ->
         map (fun theta ->comp(theta,sigma))
               (univ_unifR (comp (sigma,f''),
                            comp (sigma,g'') ) ) )
           Sol )      ;;

(* abs_occ realiza a abstraccion dun termo. Xa que esta
funcion e iterativa, recollee a sustitucion-abstraccion
previa, e as novas variables empleadas (para o final
expresar correctamente o dominio da sustitucion
abstraccion). Utiliza a abs_occ_set para realizar
iteradamente a abstraccion dos argumentos  *)


let rec abs_occ ((Subst(b',funf,b) as s),termo,novas_vars)  =
 match termo with (Var x) ->
  ( let  ((b'',(jb',jx)),univ) = coproduct( b',[1])
   in let (Subst(_,funjx,_)) = jx
   in let (Var (nova_var)) = funjx 1
  in (b'',univ(s,
        Subst([1],(fun 1 -> (subst_apply (Var x) s)
                     |_ ->  raise non_def),
              b) ), funjx 1,
        union(novas_vars,[nova_var]))
  )
|Apply(Opr(rot,srot),funrot) ->
  let  (b'',Subst',func,_,n_vars) =
     abs_occ_set(b',s,funrot,srot,novas_vars)
 in (b'',Subst',Apply(Opr(rot,srot),func),n_vars)

and abs_occ_set ((b,subt,funrot,srot,novas_vars) as bsfs) =
 if srot=[]  then bsfs
   else let  (x::s') =  srot
   in let (b',subt',termo',n_vars) = abs_occ(subt,(funrot x),
```

```
                                                    novas_vars)
   in abs_occ_set(b',subt', (fun y -> if y = x then termo'
                                  else (funrot y)) , s',n_vars)
;;

(* as reglas (axiomas) son etiquetadas co nome de 2-
celulas *)

type 'a two_Cell = Two_cell of 'a*'a  ;;

(* build_axioms construe o conxunto de 2-celulas
correspondentes a unha string dada *)

let build_axioms rules_sys =
 let  (_,(s_rule,t_rule)) = build rules_sys
     in let (Subst(a,funf,b)) = s_rule
        and (Subst(_,fung,_)) = t_rule
in
it_list
 (fun acc  -> fun x ->
  (Two_cell(Subst([1],(fun 1 -> funf x|_ ->raise non_def),b),
           Subst([1],(fun 1 -> fung  x|_ ->raise non_def),b ))
  )::acc) [] a  ;;

(* chequea a igualdade sintactica de dous termos *)

let rec  equal_term ((t1,t2)as v) =
 match (t1,t2) with (Var x, Var y) -> x = y
 |(Apply(Opr(ri,s),f),Apply(Opr(ro,t),g) ) ->
  ri = ro &
     ( (s=[] &   t=[])   or  (not(s=[] or t=[] )  &
    let  (x::s1) =  s
    in let (y::t1) =  t
  in equal_term( f x, g y) &  equal_term_set((s1,f),(t1,g))
                             )
     )
 | (_,_) -> false

and equal_term_set ((s1,f),(t1,g)) =
 (s1=[] & t1 =[]) or (not(s1=[]or t1=[]) &
  let (x::ss1) =s1 in let (y::tt1)= t1 in equal_term(f x,g y)
&
```

```
equal_term_set((ss1,f),(tt1,g)) );;

(* chequea a identidade sintactica de duas sustitucions *)

let ident_Subst (f,g) =
 let  (Subst(a,f2,b)) = f
 in let (Subst(c,g2,d)) = g
in seteq(a,c) &   seteq(b,d) &
    equal_term_set ((a,f2),(a,g2))   ;;


(* --------- FIN FASE PREVIA A E_UNIFICACION ----------- *)

(* ------ FUNCIONS QUE REALIZAN A E-UNIFICACION --------- *)
(*  As funcions definidas nesta parte son as seguintes :


rel_abs_narrow ---- Corresponde a imitacion por abstraccion
                    do metodo de Dougherty e Johann
narrow_abs -------- E a funcion que proporciona as
                    solucions polo novo metodo
rel_narr ---------- A un par de cadeas de simbolos devolve
                    o numero de solucions por rel_abs_narrow
nar_abss ---------- Igual que rel_narr para o novo metodo
rel_nar_list ------ Mostra a lista coas solucions por
                    rel_abs_narrow
nar_abs_list ------ Igual que a anterior con noso metodo

                    ---------------------------------- *)

(* rel_abs_narrow e a funcion que imita a forma de realizar
a unificacion ecuacional pola variante do metodo de
Dougherty e Johann que realiza a abstraccion posicion para
as variables tanto do problema como do lado esquerdo da
regra. Ten como argumentos unha lista de 2-celulas cos
axiomas de E, se ben estan considerados orientados (e.g falla
o unficar b = c se a teoria e a =e b y a =e c), para evitar
mais problemas de non terminacion que os implicitos no
propio problema. Esto e debido a que realmente desexamos
comparar en situacions analogas ambos metodos. *)
```

```
let rel_abs_narrow R =
 let rec rel_abs_narrowR  (f,g) =
  let  (Subst(a,fnf,b)) = f
 in if ident_Subst (f,g) then  (ident b)::[]
    else if (list_length a) = 1 then
     try (let   r = unify (f,g)
      in r::(R_Narr  (f,g))@(R_Narr  (g,f))  )
   with _ ->  (R_Narr  (f,g))@(R_Narr  (g,f))
   else if a=[] then Subst_null::[]
        else decrease  rel_abs_narrowR    (f,g)

and R_Narr  (f,g) =
 let  (Subst(a,fnf,b)) = f
 in let (x::_) =  a
 in let (b',theta,term_abs,new_vars) =
   abs_occ(ident (b),fnf x,[])
 in
  let rec apply_in_occ occ =
   fun  (Var _)  -> []
  |((Apply(Opr(ro,sro),fnro))as term) ->
   let   subst_term = Subst([1],( fun 1 ->
                          term|_ ->raise non_def) ,new_vars)
   in let  apply_rule = fun  (Two_cell(s_rule,t_rule)) ->
    let  (Subst(a_rule,fn_rule,b_rule)) = s_rule
    in let (y::_) =  a_rule
    and  ((bbrule,(jbbrule,jbruleb)),univ_bbrule) = coproduct

       (b,b_rule)
    and ((b'brule,(jb'brule,jbruleb')),univ_b'brule) =
          coproduct (b',b_rule)
    in let (b'',theta',term_rule',n_vars) =
        abs_occ(univ_b'brule(comp (jbbrule,theta),
                         jbruleb),
         (subst_apply (fn_rule y)jbruleb'  ),new_vars)
    in let theta'' = comp (theta',inclusion(n_vars,b''))
    in let h = comp (inclusion(new_vars,n_vars),subst_term)
    in let k = Subst([1],
               (fun 1 -> term_rule'|_->raise non_def),n_vars)
    in try (
       let r = unif_nrepeat_vars(h,k)
    in let (Subst(_,_,c)) = r
       and t_rule' = comp (jbruleb,t_rule)
```

```
    in let f'= replace (comp (jbbrule,f)) occ t_rule'
       and g'=comp (jbbrule,g)
       and f'' = theta''
    in let g'' = comp (f'', comp (inclusion(c,n_vars),r))
    in let ((ab'',(ja_nvars,jnvars_a)),univ_a_nvars) =
        coproduct(a,n_vars)
    in let Sol = rel_abs_narrowR (univ_a_nvars(f',f''),
                          univ_a_nvars(g',g''))
   in  map(fun tau -> comp (tau,jbbrule)) Sol
   ) with  _ -> []
  in flatlist(map(apply_rule) R) @
     flatlist(map(fun i -> apply_in_occ (occ@[i]) (fnro i))
                 sro)
 in apply_in_occ [0] term_abs
in rel_abs_narrowR  ;;

(* narrow_abs e a funcion que implementa o noso  algoritmo,
coas especificacions  indicadas tamen para
rel_abs_narrow. *)

let  narrow_abs R =
 let rec narrow_absR  (f,g) =
  let  (Subst(a,fnf,b)) = f
 in if ident_Subst (f,g) then  (ident  b)::[]
    else
     if (list_length a) = 1 then
     try(
      let  r = unify (f,g)
     in r::(Narr_Abs  (f,g))@(Narr_Abs  (g,f)) )
     with _ ->  (Narr_Abs  (f,g))@(Narr_Abs  (g,f))
     else if a=[] then Subst_null::[]
          else decrease narrow_absR    (f,g)

 and  Narr_Abs  (f,g)  =
  let  (Subst(a,fnf,b)) = f
  in let (x::_) = a
  in let (b',theta,term_abs,new_vars) =
          abs_occ(ident (b), fnf x ,[])
   in let theta' = comp (theta,inclusion(new_vars,b'))
   in
   let rec apply_in_occ occ = fun  (Var(_)) -> []
   |((Apply(Opr(ro,sro),fnro) )as term) ->
```

```
    let subst_term =
        Subst([1],(fun 1 -> term|_->raise non_def),
              new_vars)
    in let  apply_rule (Two_cell(s_rule,t_rule)) =
     let (Subst(a_rule,fn_rule,b_rule)) = s_rule
     in let ((bbrule,(jbbrule,jbruleb)),univ_bbrule) =
              coproduct (b,b_rule)
        and  ((b'brule,(jb'brule,jbruleb')),univ_b'brule) =
              coproduct (new_vars,b_rule)
     in let h = comp (jb'brule,subst_term)
        and k = comp (jbruleb',s_rule)
     in try (
        let r = unif_fact (h,k)
     in let (Subst(_,_,c)) = r
        and t_rule' = comp (jbruleb,t_rule)
     in let f'= replace(comp (jbbrule,f)) occ t_rule'
        and g'=comp (jbbrule,g)
     in let f'' = univ_b'brule( (comp (jbbrule,theta')),
                        jbruleb )
     in let g'' = comp (f'',comp (inclusion(c,b'brule), r))
     in let ((ab'_cop,(jab',jb'a)),univ_ab') =
              coproduct(a,b'brule)
     in let Sol =  narrow_absR (univ_ab'(f',f''),
                                univ_ab'(g',g''))
   in  map(fun tau -> comp (tau,jbbrule)) Sol )
   with  _ -> []

  in flatlist(map(apply_rule) R)@
     flatlist(map(fun i ->apply_in_occ (occ@[i]) (fnro i))
              sro )

 in apply_in_occ [0] term_abs
in narrow_absR  ;;
```

As funcións `rel_abs_narrow_dec` e `narrow_abs` son as que empregan a descomposición e calculan o número de nós recorridos en cada problema.

```
let rec iterar fn =
 fun  (m,Sols,Sol)  ->
  if Sol = [] then (m,Sols)
  else let (m',Sols') = fn m (hd(Sol))
in iterar fn (m',Sols@Sols',tl(Sol));;
```

```
let fst(a,_)=a;;let snd(_,b)=b;;

let rec iterar2 fnn =
 (fun  ((m,sols) as acc)->
  (fun  R ->
    if R = [] then acc
    else let (m',solss) = fnn (hd(R)) m
   in iterar2 fnn (m',sols@solss) (tl (R))
 ));;

let decrease_it (univ_unifR) (n,(f,g))  =
 let  (Subst(a,_,_)) = f
in if a=[] then (n,Subst_null::[])
   else let   (x::a') =   a
        in let  (m,Sol) = univ_unifR (n,(restrict([x],(f,g))))
        in let (f',g') = restrict(a', (f,g))
in
 iterar
 (fun mm sigma ->
   let (m',Sol') = univ_unifR  (mm,(comp(sigma,f'),
comp(sigma,g')))
 in (m',map(fun theta -> comp(theta,sigma)) Sol')
 )
(m,[],Sol)
;;

let rel_abs_narrow_dec  nn R (f,g)=
 let rec rel_abs_narrowR (n,(f,g)) =
   let (Subst(a,fnf,b)) = f  and (Subst(_,fng,_)) = g
 in if ident_Subst(f,g) then (n,(ident b)::[])
    else if  n > nn then (nn,[Fallo_num_exc])
         else if (list_length a) = 1 then
                let (x::_) = a
              in let (s,t) = (fnf x,fng x)
              in match (s,t) with
                 (Apply(Opr(ro,sro),fnro),
                  Apply(Opr(ri,sri),fnri)) ->
                   if ro = ri then
                     let (m1,S_redund) =  rel_abs_narrowR
                      (n,(Subst(sro,fnro,b),Subst(sro,fnri,b)))
                    in  let (m2,sol_unif) =
```

```
                    try(let r = unify (f,g)
                    in  (m1+1,[r]) )
                    with _ -> (m1,[])
                in let (m,Sols) = R_Narr (m2,(f,g))
                in let (m',Sols') = R_Narr(m,(g,f))
                in (m',S_redund @ sol_unif @ Sols @  Sols')
                else let (m,Sols) = R_Narr (n,(f,g))
                in let (m',Sols') = R_Narr (m,(g,f))
                in (m',Sols@Sols')
            |(_,_) -> let (m2,sol_unif)=
                        try(let r = unify(f,g)
                        in (n+1,[r]))
                        with  _ -> (n,[])
                in let (m,Sols) = R_Narr(m2,(f,g))
                in let (m',Sols') = R_Narr(m,(g,f))
                in (m', sol_unif @ Sols @  Sols')
          else if a=[] then (n,(ident b)::[] )
              else decrease_it rel_abs_narrowR   (n,(f,g))

and
R_Narr  (n,(f,g))  =
  let  (Subst(a,fnf,b)) = f
  in let (x::_) = a
  in let  (b',theta,term_abs,new_vars) =
   abs_occ(ident (b),fnf x,[])
 in
  let rec apply_in_occ occ   = fun  (Var(_)) m -> (m,[])
   |((Apply(Opr(ro,sro),fnro) )as term) m ->
     let   subst_term = Subst([1],( fun 1 ->
                      term|_ ->raise non_def) ,new_vars)
   in let  apply_rule  (Two_cell(s_rule,t_rule))  nmn =
    let  (Subst(a_rule,fn_rule,b_rule)) = s_rule
    in let (y::_) =  a_rule
    and  ((bbrule,(jbbrule,jbruleb)),univ_bbrule) =
         coproduct(b,b_rule)
    and ((b'brule,(jb'brule,jbruleb')),univ_b'brule) =
        coproduct (b',b_rule)
    in let (b'',theta',term_rule',n_vars) =
        abs_occ(univ_b'brule(comp (jbbrule,theta),
                        jbruleb),
         (subst_apply (fn_rule y)jbruleb'  ),new_vars)
    in let theta'' = comp (theta',inclusion(n_vars,b''))
```

```
    in let h = comp (inclusion(new_vars,n_vars),subst_term)
    in let k = Subst([1], (fun 1 -> term_rule'|_->raise
                                          non_def) , n_vars)
    in try (
       let r = unif_nrepeat_vars(h,k)
    in let (Subst(_,_,c)) = r
    in let t_rule' = comp (jbruleb,t_rule)
    in let f'= replace (comp (jbbrule,f)) occ t_rule'
    in let g'=comp (jbbrule,g)
    in let f'' = theta''
    in let g'' = comp (f'',   comp(inclusion(c,n_vars),r))
    in let ((ab'',(ja_nvars,jnvars_a)),univ_a_nvars) =
        coproduct(a,n_vars)
    in let (malfa,Sol) = rel_abs_narrowR
                          (nmn+1,((univ_a_nvars(f',f''),
                          univ_a_nvars(g',g''))))
    in  (malfa,map(fun tau -> comp (tau,jbbrule)) Sol ))
    with  _ -> (nmn,[])
  in let (mm,sols) = iterar2 apply_rule (m,[]) R
  in iterar2 (fun i-> apply_in_occ (occ@[i]) (fnro i))
(mm,sols) sro
 in apply_in_occ [0] term_abs n
in rel_abs_narrowR  (0,(f,g)) ;;

let narrow_abs_dec  nn R (f,g)=

 let rec narrow_absR (n,(f,g)) =
   let (Subst(a,fnf,b)) = f  and (Subst(_,fng,_)) = g
 in if ident_Subst(f,g) then (n,(ident b)::[])
    else if  n > nn then (nn,[Fallo_num_exc])
         else if (list_length a) = 1 then
                let (x::_) = a
              in let (s,t) = (fnf x,fng x)
              in match (s,t) with
                  (Apply(Opr(ro,sro),fnro),
                   Apply(Opr(ri,sri),fnri)) ->
                   if ro = ri then
                     let (m1,S_redund) =
                       narrow_absR (n,(Subst(sro,fnro,b),
                                       Subst(sro,fnri,b)))
                   in let (m2,sol_unif) =
                      try(let r =unify(f,g)
```

```
                    in  (m1+1,[r]) )
                    with _ -> (m1,[]) in
                    let (m,Sols) = Narr_Abs(m2,(f,g))
                    in let (m',Sols') = Narr_Abs(m,(g,f))
                 in (m',S_redund @ sol_unif @ Sols @  Sols')
                 else let (m,Sols) = Narr_Abs (n,(f,g))
                      in let (m',Sols') = Narr_Abs(m,(g,f))
                 in (m',Sols@Sols')
              |(_,_) -> let (m2,sol_unif)=
                          try(let r = unify(f,g) in (n+1,[r]))
                          with  _ -> (n,[]) in let (m,Sols) =
                              Narr_Abs(m2,(f,g))
              in let (m',Sols') = Narr_Abs(m,(g,f))
              in (m', sol_unif @ Sols @  Sols')

    else if a=[] then (n,(ident b)::[] )
         else decrease_it narrow_absR   (n, (f,g))

and     Narr_Abs  (n,(f,g))  =
 let  (Subst(a,fnf,b)) = f
 in let (x::_) = a
 in let (b',theta,term_abs,new_vars) = abs_occ(ident (b),fnf x ,[])
 in let theta' = comp (theta,inclusion(new_vars,b'))
 in
  let rec apply_in_occ occ   = fun  (Var(_)) m -> (m,[])
  |((Apply(Opr(ro,sro),fnro) )as term) m ->
    let subst_term = Subst([1],(fun 1 -> term|_->raise
                                        non_def), new_vars)
    in let  apply_rule (Two_cell(s_rule,t_rule))  nmn =
     let (Subst(a_rule,fn_rule,b_rule)) = s_rule
     in let ((bbrule,(jbbrule,jbruleb)),univ_bbrule) =
            coproduct   (b,b_rule)
     and  ((b'brule,(jb'brule,jbruleb')),univ_b'brule) =
            coproduct   (new_vars,b_rule)
     in let h = comp (jb'brule,subst_term)
     in let k = comp (jbruleb',s_rule)
     in try (
        let r = unif_fact (h,k)
     in let (Subst(_,_,c)) = r
     in let t_rule' = comp (jbruleb,t_rule)
     in let f'= replace(comp (jbbrule,f)) occ t_rule'
     in let g'=comp (jbbrule,g)
```

```
      in let f'' = univ_b'brule( (comp (jbbrule,theta')),
                        jbruleb )
      in let g'' = comp (f'',  comp (inclusion(c,b'brule),r))

      in let ((ab'_cop,(jab',jb'a)),univ_ab') =
         coproduct(a,b'brule)
      in let (malfa,Sol) =
           narrow_absR (nmn+1,(univ_ab'(f',f''),univ_ab'(g',g'')))
      in  (malfa,map(fun tau -> comp (tau,jbbrule)) Sol ))

    with  _ -> (nmn,[])
   in let (mm,sols) = iterar2 apply_rule (m,[]) R
   in iterar2 (fun i-> apply_in_occ (occ@[i]) (fnro i))(mm,sols) sro

 in apply_in_occ [0] term_abs n
in narrow_absR  (0,(f,g)) ;;


(* rel_narr da terna formada por : o conxunto de 2-
celulas, o diccionario previo (coas adxudicacions das
variables), e o problema dado, mostra o numero de nos
procesados e o conxunto coas solucions resultantes.
O primeiro argumento de rel_abs_narrow e un limite do numero
maximo de nos recorridos *)


let rel_narr (R,D,P) =
let (n,Sols)  = rel_abs_narrow 10000 R P
in  print_string "numero de nodos "; print_int n; output_Sol D
Sols;;

(* realiza a mesma operacion respecto de narrow_abs *)

let nar_abss (R,D,P) =
let (n,Sols)  = narrow_abs 10000 R P
in  print_string "numero de nodos "; print_int n; output_Sol D
Sols;


#
let R = build_axioms "{a=b,g(h(a),h(b),x,x)= d}" ;; (* presentacion *)

let (D,P) = build "{g(x,x,h(a),h(b)) = d}";;  (* problema *)
```

```
rel_narr (R,D,P);;

R : substitution two_Cell list = [Two_cell (Subst ([1], <fun>,
[1; 1]), Subst ([1], <fun>, [1; 1])); Two_cell (Subst ([1],
<fun>, [1; 1]), Subst ([1], <fun>, [1; 1]))]
#P : substitution * substitution = Subst ([1], <fun>, [1; 1]),
Subst ([1], <fun>, [1; 1])
D : int -> string = <fun>
#numero de nodos 21[1  SOLUCION
   x -> h(a()) ,
   x -> h(a())
2  SOLUCION
   x -> h(a()) ,
   x -> h(a())
3  SOLUCION
   x -> h(b()) ,
   x -> h(b())
4  SOLUCION
   x -> h(b()) ,
   x -> h(b())
5  SOLUCION
   x -> h(a()) ,
   x -> h(a())
6  SOLUCION
   x -> h(b()) ,
   x -> h(b())]
- : unit = ()

nar_abss(R,D,P);;
numero de nodos 8[1  SOLUCION
   x -> h(a()) ,
   x -> h(a())
2  SOLUCION
   x -> h(b()) ,
   x -> h(b())]
- : unit = ()
```

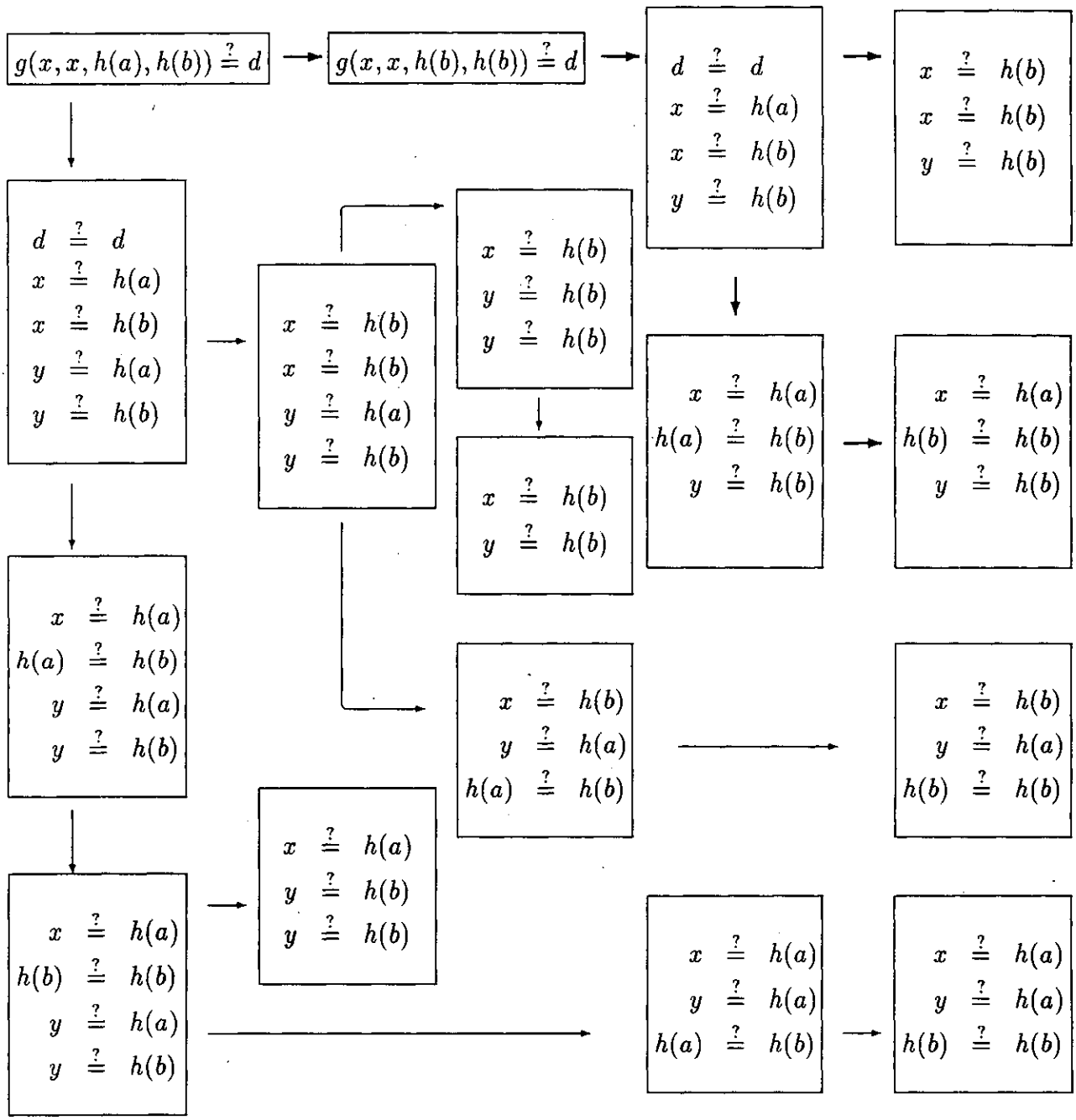
Apéndice B

Resultados Comparativos

Empregando os programas do Apéndice A, realizóuse un estudio comparativo entre os métodos de *Relaxed Narrowing* e o *narrowing con abstracción*. Por outra parte, se estudia con certo detalle o número e forma producir solucións de *Relaxed Paramodulation* e de *Narrowing con Abstracción*, usando como exemplo o que aparece resolto na páxina 213. As solucións producidas por `narrow_abs` son as que corresponden as líneas de trazo groso, mentras que as que se xeran por `rel_abs_narrow` son tanto as de trazo fino como as de trazo groso. Preséntanse tamén 4 taboas coas seguientes medidas nuna serie de experimentos realizados na estación de traballo Sun Sparc IPX do Departamento de Computación.

- Taboa B.1. Problemas para os que a función `rel_abs_narrow` non finaliza e sí que o fai `narrow_abs`
- Taboa B.2 Resultados experimentales co número de soluciones proporcionadas por `rel_abs_narrow_dec`, `narrow_abs_dec`, `rel_abs_narrow` e `narrow_abs`
- Taboa B.3 Comparación entre o número de nós procesados empregando as funciones anteriores
- Taboa B.4 Tempo invertido na resolución dos problemas anteriores para `rel_abs_narrow` e `narrow_abs`

Exemplo B.1


$g(x,x,h(a),h(b)) \stackrel{?}{=} d$
$g(x,x,h(b),h(b)) \stackrel{?}{=} d$
$d \stackrel{?}{=} d$
$x \stackrel{?}{=} h(a)$
$x \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$h(a) \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$h(b) \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$h(a) \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(a)$
$h(a) \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(b)$
$h(b) \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$h(b) \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(a)$
$h(b) \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$y \stackrel{?}{=} h(a)$
$h(a) \stackrel{?}{=} h(b)$
$x \stackrel{?}{=} h(a)$
$h(b) \stackrel{?}{=} h(b)$
$y \stackrel{?}{=} h(b)$

O código empleado es el siguiente :

- Nós : para o número de nós da árbore de derivacións producidas,
- Sols : indica o número de solucións producidas por cada procedemento,
- NA : para indicar que se tén realizado mediante `narrow_abs`,
- NAD : para indicar que foi obtido empregando `narrow_abs_dec`,
- RN : para indicar que se ten realizado mediante `rel_abs_narrow`
- RND : para indicar se usou para elo a función `rel_abs_narrow_dec`.

En ciertos casos a función `rel_abs_narrow` non remata, e sí que o fai a función `narrow_abs`. Como exemplo de ello :

Ecuacións	*Sistema*	*Sols NA*	*Sols RN*
$\{f(x,x) \Rightarrow f(a,b)\}$	$f(x,x) \stackrel{?}{=} f(a,b)$	1	no finaliza
$\{f(x,x) \Rightarrow f(a,b)\}$	$f(x,x) \stackrel{?}{=} d$	0	no finaliza

Taboa B.1 Problemas nos que un deles non remata

Ecuacions	*Sistema*	*Sols NA*	*Sols NAD*	*Sols RN*	*Sols RND*
$\{a \Rightarrow b,$ $g(x,x,a,b) \Rightarrow d\}$	$g(a,b,x,x) \stackrel{?}{=} d$	2	2	6	6
$\{a \Rightarrow b,$ $g(x,x) \Rightarrow h(x,x)\}$	$g(a,b) \stackrel{?}{=} h(a,b)$	3	5	5	14
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	28	28	44	45
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	133	133	211	212
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$x \stackrel{?}{=} p(e,in(x))$ $e \stackrel{?}{=} p(e,in(x))$	21	21	40	40
$\{h(x,x) \Rightarrow x$ $f(a,x) \Rightarrow g(x,b)\}$ $g(x,c) \Rightarrow f(c,x)$	$h(f(x,y),g(x,y))$ $\stackrel{?}{=} g(x,y)$	4	7	7	26
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(a,y,z),g(x_2,z)),$ $g(a,a)) \stackrel{?}{=} g(a,a)$	1	2	5	20
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $g(u,v)) \stackrel{?}{=} h(u_2,v_2)$	55	87	669	2344
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(f(x,y,z),g(x_2,z))$ $\stackrel{?}{=} h(u_2,v_2)$	23	26	58	137
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $f(b,u,z)) \stackrel{?}{=} f(b,u,z)$	0	0	0	0

Taboa B.2 Comparación entre o número de solucións

Ecuaciones	*Sistema*	*Nos NA*	*Nos NAD*	*Nos RN*	*Nos RND*
$\{a \Rightarrow b,$ $g(x,x,a,b) \Rightarrow d\}$	$g(a,b,x,x) \stackrel{?}{=} d$	8	8	21	21
$\{a \Rightarrow b,$ $g(x,x) \Rightarrow h(x,x)\}$	$g(a,b) \stackrel{?}{=} h(a,b)$	11	19	18	37
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	141	141	207	208
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	688	688	1027	1028
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$x \stackrel{?}{=} p(e,in(x))$ $e \stackrel{?}{=} p(e,in(x))$	80	80	147	147
$\{h(x,x) \Rightarrow x$ $f(a,x) \Rightarrow g(x,b)\}$ $g(x,c) \Rightarrow f(c,x)$	$h(f(x,y),g(x,y))$ $\stackrel{?}{=} g(x,y)$	58	64	113	173
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(a,y,z),g(x_2,z)),$ $g(a,a)) \stackrel{?}{=} g(a,a)$	9	11	39	78
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $g(u,v)) \stackrel{?}{=} h(u_2,v_2)$	368	507	4342	9325
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(f(x,y,z),g(x_2,z))$ $\stackrel{?}{=} h(u_2,v_2)$	134	147	344	569
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $f(b,u,z)) \stackrel{?}{=} f(b,u,z)$	7	7	24	28

Taboa B.3. Comparación entre o número de nós recorridos.

A continuación móstrase o tempo de resolución dos problemas empregados para a comparación :

Ecuacions	*Sistema*	*NA*	*RN*
$\{a \Rightarrow b,$ $g(x,x,a,b) \Rightarrow d\}$	$g(a,b,x,x) \stackrel{?}{=} d$	$0.6s$	$0.9s$
$\{a \Rightarrow b,$ $g(x,x) \Rightarrow h(x,x)\}$	$g(a,b) \stackrel{?}{=} h(a,b)$	$0.7s$	$1.3s$
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	$5.4s$	$12.4s$
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$p(p(x,e),in(p(x,y)) \stackrel{?}{=} e$	$32s$	$1min14s$
$\{p(x,in(x)) \Rightarrow e$ $p(x,e) \Rightarrow x\}$ $in(e) \Rightarrow e$ $p(x,x) \Rightarrow x$	$x \stackrel{?}{=} p(e,in(x))$ $e = p(e,in(x))$	$6.2s$	$9.5s$
$\{f(x,x) \Rightarrow x$ $f(a,x) \Rightarrow g(x,b)\}$ $g(x,c) \Rightarrow f(c,x)$	$h(f(x,y),g(x,y)) \stackrel{?}{=} g(x,y)$	$7.5s$	$24.4s$
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(a,y,z),g(x_2,z)),$ $g(a,a)) \stackrel{?}{=} g(a,a)$	$5.7s$	$10s$
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $g(u,v)) \stackrel{?}{=} h(u_2,v_2)$	$50s$	$15min37s$
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(f(x,y,z),g(x_2,z))$ $\stackrel{?}{=} h(u_2,v_2)$	$38.2s$	$43s$
$\{f(a,x,x) \Rightarrow g(a,x)$ $h(x,x) \Rightarrow x\}$	$h(h(f(x,y,z),g(x_2,z)),$ $f(b,u,z)) \stackrel{?}{=} f(b,u,z)$	$8.2s$	$10.4s$

Taboa B.4. Tempo de resolución dos problemas.

Indice analítico