Sistemi di Riscrittura per Termini del Prim'Ordine

Paola Inverardi, [♡] Monica Nesi [♡] e Marisa Venturini Zilli [◇]

[♡] Dipartimento di Matematica Pura e Applicata
Università di L'Aquila

[◇] Dipartimento di Scienze dell'Informazione
Università di Roma 'La Sapienza'

Rapporto Tecnico No. 35, Luglio 1999
Dipartimento di Matematica Pura e Applicata
Università degli Studi di L'Aquila

L'elaborazione di queste note è stata svolta nell'ambito della collaborazione tra l'Università di L'Aquila e l'Università 'La Sapienza' di Roma, grazie al supporto finanziario del progetto MURST 40%
"Tecniche formali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di sistemi software".

Indice

1 Introduzione

Un *sistema di riscrittura di termini* è un particolare sistema di riduzione astratto $\langle A, \longrightarrow \rangle$ [12], in cui A è l'insieme $\mathcal{T}(\Sigma, V)$ dei termini del prim'ordine con simboli di funzione in Σ e variabili in V.

Nel seguito verranno introdotte le nozioni basilari della riscrittura dei termini. In particolare verranno trattati il completamento di una teoria equazionale E e l'E-unificazione di termini. Per maggiori dettagli il lettore può fare riferimento a [1, 3].

2 Termini del prim'ordine

Definiamo l'insieme dei termini di un linguaggio del prim'ordine senza *sorte*, in quanto tale semplificazione fa conservare tutte le nozioni che interessano.

Dato un insieme Σ di simboli di funzione, che chiameremo *segnatura*, definiamo i termini su tale segnatura come tutte le espressioni legittime costruibili a partire dai simboli di funzione in Σ. Se inoltre aggiungiamo un insieme numerabile di variabili, otteniamo l'insieme dei termini su Σ e con variabili in V.
Il numero $n \in \mathbb{N}$ degli argomenti di un simbolo di funzione $f \in \Sigma$ viene detto *arità* di f. Se $n = 0$, f è un simbolo di *costante*. Con Σ^n indichiamo l'insieme di tutti i simboli in Σ di arità n. Quindi si ha $\Sigma = \bigcup_n \Sigma^n$.

Definizione 1 *Siano Σ una segnatura finita e V un insieme numerabile di variabili. L'insieme $Ter = \mathcal{T}(\Sigma, V)$ dei termini (finiti) con simboli di funzioni in Σ e variabili in V è definito come segue:*

i) $f \in Ter$ *per ogni* $f \in \Sigma^0$
ii) $V \subseteq Ter$
iii) $f(t_1, \ldots, t_n) \in Ter$ *per ogni* $f \in \Sigma^n$ *e* $t_i \in Ter$, $i = 1, \ldots, n$.

L'uguaglianza in Ter, notazione $=$, è l'identità sintattica.
Nel seguito useremo indifferentemente Ter, $\mathcal{T}(\Sigma, V)$ o $Ter(\Sigma, V)$ per denotare l'insieme dei termini su una segnatura Σ e con variabili in V. Con $Var(t)$ indichiamo l'insieme delle variabili che occorrono nel termine t. Se $Var(t) = \emptyset$ allora t è detto *chiuso*. L'insieme dei termini chiusi è denotato da Ter_0, con $Ter_0 \subset Ter$, oppure da $\mathcal{T}(\Sigma)$.
Con $\Sigma(t)$ indichiamo l'insieme dei simboli di Σ che occorrono nel termine t.

Esempio 1: La segnatura sia:

$$\Sigma = \{0, succ, =\}$$

con arità rispettivamente 0, 1, 2.
I termini $succ(x)$, $succ(succ(y))$ sono esempi di termini del particolare Ter su tale segnatura. I termini 0, $succ(0)$, $succ^2(0)$ sono in Ter_0. •

Osservazione 1: L'insieme V e la relazione $=$ sono in ogni linguaggio del prim'ordine per cui vengono anche detti *simboli logici*.
I termini col simbolo $=$, ad esempio $=(x, y)$, sono considerati termini *booleani*. Questo vale anche per altri simboli di predicato.
Va notato che a volte le variabili, ossia i simboli in V, vengono (impropriamente) incluse nella segnatura. In ogni caso deve sempre aversi $V \cap \Sigma = \emptyset$, condizione sottintesa nel seguito. ■

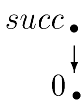
Osservazione 2: La denominazione 'del prim'ordine' vuol esprimere che le variabili (libere) sono individuali, ossia sono sostituibili da qualsiasi termine; semanticamente sono interpretabili soltanto come individui (e non, ad esempio, come insieme di individui) del dominio semantico dove si interpreta il linguaggio. Ovviamente, in un linguaggio del prim'ordine che descrive insiemi, le variabili vanno interpretatate come insiemi e saranno gli insiemi di insiemi a non essere denotati in tale linguaggio. ■

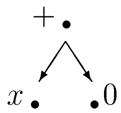
Nel seguito occorrerà trattare la struttura dei termini. Iniziamo col rappresentarli come alberi.

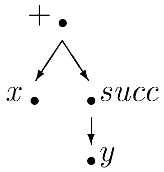
2.1 Rappresentazione mediante alberi ordinati etichettati

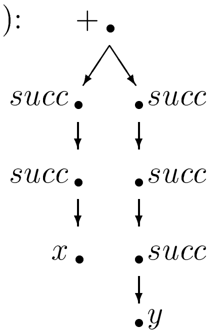
Ogni elemento $t \in Ter$ può essere rappresentato in modo naturale come un albero finito ordinato etichettato ai nodi con elementi di $\Sigma(t)$ e di $Var(t)$. Le variabili possono etichettare soltanto nodi foglie dell'albero, eccetto quando il termine coincide con una variabile (in tal caso la foglia dell'albero coincide con la radice). Come esempio mostriamo alcuni termini con i rispettivi alberi.

- 0: $\quad 0_{\bullet}$
- x: $\quad x_{\bullet}$

- $succ(0)$: 

- $+(x, 0)$: 

- $+(x, succ(y))$: 

- $+(succ^2(x), succ^3(y))$: 

Puntualizziamo che tali alberi sono a ramificazione finita visto che l'arità è sempre finita e il numero dei rami uscenti da ogni nodo esprime appunto l'arità del simbolo che lo etichetta. Le foglie dell'albero sono quindi etichettate da variabili oppure da costanti.
Gli alberi sono ordinati in quanto il termine $f(x, y)$ è diverso da $f(y, x)$, essendo l'uguaglianza tra termini di Ter l'identità sintattica. Denominiamo $Alberi_{Ter}$, con $T \in Alberi_{Ter}$, l'insieme di alberi che corrisponde a Ter, con $t \in Ter$.

2.2 Sottotermini, occorrenze, posizioni, contesti

Sottotermine di un termine t è il termine che corrisponde ad un sottoalbero dell'albero T che corrisponde a t. Ad esempio $succ(0)$ è un sottotermine di

$+(succ(0), x)$. Per indicare che s è un sottotermine di t scriviamo semplicemente $s \in t$. La nozione di sottotermine va tenuta distinta da quella di *occorrenza*. Infatti un sottotermine può avere piú di una occorrenza dentro lo stesso termine, come $succ(0)$ in $+(succ(0), succ(0))$. $Os(t)$ è il multinsieme di tutte le occorrenze di s in t. Qualora $Os(t) = \emptyset$, s non è sottotermine di t. $O(t)$ è il multinsieme di tutte le occorrenze di tutti i sottotermini di t. Per individuare una particolare occorrenza di un sottotermine risulta utile usare le posizioni, mentre per indicare semplicemente che un termine è sottotermine di un altro bastano i contesti come passiamo a precisare.

Introduciamo dapprima la nozione di *posizione* (detta a volte anche occorrenza appunto perché individua in modo univoco una determinata occorrenza).

Definizione 2 *Dato $t \in \mathcal{T}(\Sigma, V)$, l'insieme $Pos(t)$ delle* posizioni *di t è un insieme di stringhe di numeri interi positivi, definito induttivamente come segue:*

i) se $t \in V$ allora $Pos(t) = \{\epsilon\}$ dove ϵ denota la stringa vuota
ii) se $t = f(t_1, \ldots, t_n)$ per $f \in \Sigma^n$ e $t_i \in \mathcal{T}(\Sigma, V)$, allora
$Pos(t) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{i.p \mid p \in Pos(t_i)\}$
dove . denota l'operazione di concatenazione di stringhe.

Le posizioni saranno denotate con p, q, p', etc. L'insieme delle posizioni è parzialmente ordinato mediante l'*ordinamento del prefisso* definito nel modo seguente: date due posizioni p e q, si ha che $p \leq q$ se e solo se esiste una posizione p' tale che $p.p' = q$. In tal caso si dice che p è *prefisso* di q. Se due posizioni p e q non sono confrontabili rispetto all'ordinamento del prefisso, allora si dice che p e q sono *parallele* o *disgiunte*, indicando ciò con $p \parallel q$.

Quindi, una posizione in un termine t è una stringa di interi che denota il cammino di accesso ad una occorrenza di un sottotermine s nell'albero T di t; piú precisamente, una posizione in un termine t è una stringa di interi che denota il cammino di accesso al nodo che è radice del sottoalbero che corrisponde ad una occorrenza di un sottotermine s di t.

Definizione 3 *Dati $t \in \mathcal{T}(\Sigma, V)$ e $p \in Pos(t)$, l'occorrenza alla posizione p, ossia il* sottotermine *di t che occorre alla posizione p, denotato da $t|_p$, è definito per induzione sulla lunghezza di p:*

$$\begin{aligned} t|_\epsilon &= t \\ f(t_1, \ldots, t_n)|_{i.q} &= t_i|_q \end{aligned}$$

oppure, equivalentemente

$$t|_p = \begin{cases} t & se\ p = \epsilon \\ t_i|_q & se\ p = i.q\ \ e\ t = f(t_1, \ldots, t_i, \ldots, t_n) \end{cases}$$

Definizione 4 *Dati* $t, t' \in \mathcal{T}(\Sigma, V)$ *e* $p \in Pos(t)$*, il* termine ottenuto da t sostituendo il sottotermine alla posizione p con t'*, denotato da* $t[t']_p$*, è definito per induzione sulla lunghezza di* p*:*

$$\begin{aligned} t[t']_\epsilon &= t' \\ f(t_1, \ldots, t_n)[t']_{i.q} &= f(t_1, \ldots, t_i[t']_q, \ldots, t_n) \end{aligned}$$

Infatti, se $p = i.q$ si ha che il termine t è della forma $f(t_1, \ldots, t_n)$ per un qualche simbolo di funzione $f \in \Sigma^n$, $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, V)$ e $1 \leq i \leq n$.

Come abbiamo già detto, a volte p viene usata come $t|_p \in O(t)$.
Talvolta occorre distinguere tra posizioni di variabili e posizioni non di variabili. L'insieme delle *posizioni non di variabili* di un termine $t \in \mathcal{T}(\Sigma, V)$ è dato da

$$Pos'(t) = \{\ p \in Pos(t) \mid t|_p \notin V\ \}.$$

Esempio 2: Siano $a, f, g \in \Sigma$ e $x, y \in V$. Dato il termine $t = f(x, g(y, a, a))$, il suo insieme di posizioni è $Pos(t) = \{\epsilon, 1, 2, 2.1, 2.2, 2.3\}$ ed i sottotermini sono: $t|_\epsilon = f(x, g(y, a, a))$, $t|_1 = x$, $t|_2 = g(y, a, a)$, $t|_{2.1} = y$, $t|_{2.2} = a$, $t|_{2.3} = a$. Le due occorrenze della costante a risultano in tal modo univocamente individuate. L'insieme delle posizioni non di variabili è $Pos'(t) = \{\epsilon, 2, 2.2, 2.3\}$.

•

Qualora non sia necessario precisare la posizione di una occorrenza volendo semplicemente dire che il termine s è un sottotermine del termine t, basta usare la notazione contesto, vale a dire scrivere $t = C[s]$, dove $C[\ ]$ è detto appunto *contesto* ed è ottenuto da un termine mediante la cancellazione di una occorrenza di un sottotermine. Il simbolo $[\ ]$ sta per l'occorrenza vuota. Ad esempio, in $f(g(a), a) = C[a]$, $C[\ ]$ è sia $f(g(a), [\ ])$ che $f(g([\ ]), a)$.
Viene anche usato $C[a, a]$ per evidenziare entrambe le occorrenze di a, facendo uso del contesto 'doppio' $C[\ ,\ ] = f(g([\ ]), [\ ])$. In generale per evidenziare n occorrenze (senza precisarne la posizione) viene usato un contesto con n occorrenze vuote.
Di frequente uso risulta anche $t[a]$ in luogo di $t = C[a]$.
Volendo precisare la posizione usando la notazione contesto, possiamo usare la notazione ridondante, spesso usata, $C[\ ]|_p$ e quindi $C[s]|_p$ per denotare il sottotermine s alla posizione p.

2.3 Sostituzioni, rimpiazzamenti

Le variabili si caratterizzano per il fatto di poter essere sostituite tramite le sostituzioni.

Definizione 5 *Siano Σ una segnatura e V un insieme infinito numerabile di variabili. Una* sostituzione *è una funzione $\sigma : V \to \mathcal{T}(\Sigma, V)$ tale che $\sigma(x) \neq x$ solo per un numero finito di variabili x. Per ogni sostituzione σ definiamo:*
i) il dominio *$\mathcal{D}(\sigma)$, come l'insieme (finito) delle variabili sostituite da σ:*

$$\mathcal{D}(\sigma) = \{x \in V \mid \sigma(x) \neq x\} \ , \ \ \sigma(x) \in \mathcal{T}(\Sigma, V)$$

*ii) l'*insieme *$\mathcal{I}(\sigma)$* delle variabili introdotte da *σ:*

$$\mathcal{I}(\sigma) = \bigcup_{x \in \mathcal{D}(\sigma)} Var(\sigma(x)).$$

Quindi, una sostituzione può essere rappresentata da un insieme finito di legami del tipo $\{\sigma(x)/x \mid x \in \mathcal{D}(\sigma)\}$. Ogni sostituzione $\sigma : V \to \mathcal{T}(\Sigma, V)$ può essere estesa ad una funzione $\sigma' : \mathcal{T}(\Sigma, V) \to \mathcal{T}(\Sigma, V)$ nel modo seguente: per $x \in V$ si ha $\sigma'(x) = \sigma(x)$ e per un termine non variabile $t = f(t_1, \ldots, t_n)$ si ha $\sigma'(t) = f(\sigma'(t_1), \ldots, \sigma'(t_n))$.

Una sostituzione σ si dice *chiusa* se e solo se $Var(\sigma(x)) = \emptyset$ per ogni $x \in \mathcal{D}(\sigma)$. Indicheremo con id la sostituzione identica o vuota, ossia quella per cui $\mathcal{D}(id) = \emptyset$. La composizione di due sostituzioni σ e θ è la funzione $\theta \circ \sigma$ definita come $(\theta \circ \sigma)(x) = \theta(\sigma(x))$ per ogni x, che è ancora una sostituzione: l'insieme delle sostituzioni è chiuso per composizione.

In alcuni contesti, come vedremo in seguito, verrà richiesta la condizione di *non ciclicità* per le sostituzioni, ossia $x \notin Var(\sigma(x))$ per ogni $x \in \mathcal{D}(\sigma)$. Si ha in tal caso una sostituzione *idempotente*.

Definizione 6 *Una sostituzione* idempotente *è ogni sostituzione σ tale che nessun elemento in $\mathcal{D}(\sigma)$ è sottotermine di un elemento del codominio di σ.*

Le sostituzioni idempotenti non sono chiuse per composizione.
Applicando sostituzioni idempotenti a un termine in Ter si ottiene sempre un termine in Ter, ossia non diviene un termine infinito.

2.4 Ordinamento istanziativo sui termini, equivalenza indotta

Definizione 7 *Siano s e t due termini in $\mathcal{T}(\Sigma, V)$. Si dice che s* sussume *t, e si scrive $s \leq t$, se e solo se esiste una sostituzione σ tale che $\sigma(s) = t$. Se tale σ esiste, σ viene detta una* sostituzione istanziatrice *o* match *(dall'inglese* matching substitution*) di s e t. Il termine t è un'*istanza *del termine s.*

L'ordinamento parziale sui termini può essere esteso alle sostituzioni.

Definizione 8 *Date due sostituzioni σ, σ', $\sigma \leq \sigma'$ se e solo se per ogni $x \in V$ si ha $\sigma(x) \leq \sigma'(x)$.*

Qui consideriamo sostituzioni idempotenti.

Definizione 9 *Dati due termini s e t in $\mathcal{T}(\Sigma, V)$, s e t sono* unificabili *se e solo se esiste una sostituzione σ tale che $\sigma(s) = \sigma(t)$. Se tale σ esiste, σ viene detta una* sostituzione unificatrice *di s e t.*

L'insieme di tutte le sostituzioni unificatrici di due termini s e t viene indicato con $\mathcal{U}_{\emptyset}(s, t)$.
In generale possono esistere varie sostituzioni unificatrici di due termini, oppure nessuna.

Definizione 10 *Dati due termini s e t in $\mathcal{T}(\Sigma, V)$, se s e t sono unificabili, allora esiste una unificatrice minima di s e t, detta l'*unificatrice piú generale *(abbreviata in* mgu *dall'inglese* most general unifier*). Ovvero, esiste $\sigma \in \mathcal{U}_{\emptyset}(s, t)$ tale che per ogni $\theta \in \mathcal{U}_{\emptyset}(s, t)$ si ha $\sigma \leq \theta$.*

La mgu è unica a meno di ridenominazioni di variabili e può essere calcolata con l'algoritmo descritto in Sezione 2.5.2. Infine, dalle definizioni precedenti segue che, nel caso in cui sia soddisfatta la condizione di non ciclicità per le sostituzioni, il matching tra termini è un'unificazione in una sola direzione. Infatti, un match è una sostituzione che, applicata ad *uno* solo dei due termini, li rende uguali, mentre un'unificatrice è una sostituzione che, applicata ad *entrambi* i termini, li rende uguali. Nel caso di sostituzioni generiche, quindi eventualmente cicliche, ciò non è vero in generale.

Esempio 3: Siano $a, b, f, g \in \Sigma$ e $x, y \in V$. I termini $s = f(x, a)$ e $t = f(b, y)$ sono unificabili con mgu $\sigma = \{b/x, a/y\}$, mentre non possono essere resi uguali tramite match, ovvero nessuno dei due termini sussume l'altro. I termini $s' = x$ e $t' = g(x)$ non sono unificabili, mentre possono essere resi uguali tramite la sostituzione istanziatrice (ciclica) $\sigma' = \{g(x)/x\}$ applicata ad s', ovvero $\sigma'(s') = t'$. •

2.5 Algoritmi di manipolazione simbolica sui termini

2.5.1 Istanziazione

Come abbiamo visto, il problema dell'istanziazione è sostanzialmente quello della soluzione di una disequazione $s \leq t$. Nel caso di s, $t \in Ter$, la soluzione una volta sostituita in s produce il termine t, il termine *istanziato*, ovvero ottenuto da s mediante una sostituzione istanziatrice σ in quanto $\sigma(s) = t$. Quando tale σ esiste, essa è unica. Se t risulta chiuso, la sostituzione σ viene detta anch'essa chiusa.
In $s \leq t$, t viene detto *istanza* di s ed s viene detto *antistanza* di t, potendosi passare da t ad s mediante un rimpiazzamento di una occorrenza di un sottotermine diverso da variabile di t con una variabile. Per ogni termine t risultano definiti i seguenti due sottoinsiemi in $(Ter, \leq)$:
- l'insieme *Inst(t)* di tutte le istanze, finito, infinito o vuoto, e
- l'insieme *Antinst(t)* di tutte le antistanze, sempre finito e non vuoto. Infatti, una variabile è antistanza di ogni termine.

2.5.2 Unificazione

Il problema dell'unificazione è sostanzialmente quello della soluzione di un'equazione $s = t$. Nel caso di s, $t \in Ter$, la soluzione una volta sostituita in entrambi i termini s e t produce uno stesso termine in entrambi, il termine *unificato*. Si parla in tal caso di *unificazione sintattica*, contrapposta all'*unificazione semantica* che vedremo piú avanti.

Il problema dell'unificazione è in generale indecidibile per un qualunque linguaggio. Risulta decidibile per i linguaggi del prim'ordine, ossia per Ter, e vari algoritmi sono stati formulati che differiscono per l'efficienza. Riportiamo l'*Algoritmo della forma risolta*, noto anche come *Algoritmo di Martelli-Montanari*.

Definizione 11 *Un sistema di equazioni è detto essere* in forma risolta *se ha la seguente forma:*

$$\begin{array}{rcl} x_1 & = & t_1 \\ & \dots & \\ x_n & = & t_n \end{array}$$

dove $i \neq j$ *implica* $x_i \neq x_j$, $i,j = 1, \dots, n$, *e ogni* x_i *ha occorrenza unica nel sistema (quella evidenziata come lato sinistro dell'equazione).*

Va notato che un sistema in forma risolta è un modo di rappresentare la sostituzione unificatrice idempotente $\{t_1/x_1, \dots, t_n/x_n\}$ che, applicata ai due termini da unificare, ottiene il termine unificato. L'idea dell'algoritmo è quindi quella di trovare la soluzione (una sostituzione) di un sistema di equazioni E, se esiste, cercando di trasformare detto sistema in uno in forma risolta, E_{ris}, vuoto all'inizio del procedimento. Ciò viene effettuato applicando in modo non deterministico, una alla volta, le regole date in Fig. 1 al fine di ottenere, ad algoritmo terminato, la successione

$$(E, \emptyset), \dots, (E', E'_{ris}), \dots, (\emptyset, E_{ris}).$$

(**Cancellazione**) $\dfrac{(E \cup \{t = t\}, E_{ris})}{(E, E_{ris})}$

(**Decomposizione**) $\dfrac{(E \cup \{f(t_1, \dots, t_n) = f(s_1, \dots, s_n)\}, E_{ris})}{(E \cup \{t_1 = s_1, \dots, t_n = s_n\}, E_{ris})}$

(**Fallimento 1**) $\dfrac{(E \cup \{f(t_1, \dots, t_n) = g(s_1, \dots, s_n)\}, E_{ris}), f \neq g}{\mathit{Fallimento\ e\ terminazione}}$

(**Fallimento 2**) $\dfrac{(E \cup \{x = t[x]\}, E_{ris}), x \in Var(t)}{\mathit{Fallimento\ e\ terminazione}}$

(**Elimin. di variabile**) $\dfrac{(E \cup \{x = t\}, E_{ris}), x \notin Var(t)}{(\{t/x\}E, E_{ris} \cup \{x = t\})}$

Figura 1: Regole per la forma risolta.

Osservazione 3: La sostituzione $\theta = E_{ris}$ eventualmente ottenuta dall'algoritmo dopo la sua terminazione con successo, ossia quando E è divenuto $\emptyset$ senza fallimento, è idempotente, ossia $\theta(\theta(x)) = \theta(x)$ per ogni x. Ciò deriva dal fatto che ogni variabile lato sinistro di un'equazione ha occorrenza unica nel sistema in forma risolta. ∎

Proposizione 1 1. *L'algoritmo della forma risolta termina.*
2. *L'algoritmo fornisce un sistema di equazioni in forma risolta oppure (aut) dichiara fallimento.*

Dimostrazione
1. Consideriamo la terna (n, n', n''), con n, n', $n'' \in \mathbb{N}$ tali che

- n è il numero delle variabili non risolte che occorrono nel sistema corrente E;
- n' è il numero delle occorrenze dei simboli di funzione nel sistema corrente E;
- n'' è il numero delle equazioni $t_1 = t_2$ con $t_1 \notin V$ che appartengono al sistema corrente E.

Definendo $(n, n', n'') > (n_1, n_1', n_1'')$ se
- $n > n_1$ oppure
- $n = n_1$ e $n' > n_1'$ oppure
- $n = n_1$, $n' = n_1'$ e $n'' > n_1''$,
si ha che $\{(n, n', n''), >\}$ risulta parzialmente ordinato e benfondato avendo come minimo (0, 0, 0). L'algoritmo determina un insieme $\{(n, n', n''), >\}$ che risulta totalmente ordinato in quanto la successione che ne deriva a partire da E iniziale é finita, decrescente e con minimo.
Infatti, osservando le regole, con esclusione ovviamente di quelle che provocano una fermata con fallimento, dopo una loro applicazione si ha:
- Cancellazione: $n > n_1$ se t non è chiuso in $t = t$, altrimenti $n' > n_1'$.
- Decomposizione: $n = n_1$ e $n' > n_1'$.
- Eliminazione di variabile: $n > n_1$ se la variabile risolta occorre in E, altrimenti $n = n_1$, $n' = n_1'$ e $n'' > n_1''$.

2. La terminazione senza fallimento fornisce $(\emptyset, E_{ris})$, ossia un sistema di equazioni in forma risolta (altrimenti qualche regola sarebbe ancora applicabile). ∎

Proposizione 2 *L'algoritmo della forma risolta è corretto.*

Dimostrazione Vediamo dapprima la correttezza in caso di fallimento. Osservando le due regole relative:
- (Fallimento 1): si ha conflitto di radice, che nessuna sostituzione può risolvere.
- (Fallimento 2): si ha verifica positiva della occorrenza della variabile lato sinistro dell'equazione nel termine lato destro della stessa equazione. Allora nessuna sostituzione può far ottenere un termine finito unificato, ossia un termine unificato in Ter, a meno che non si applichi la sostituzione $\{t[x]/x\}$ un numero infinito di volte, ottenendo in tal modo un termine infinito, per cui si uscirebbe da Ter.
Dalla correttezza dei casi di fallimento ne deriva quella della risposta positiva in quanto non vi sono altri casi di fallimento e l' algoritmo termina. ∎

Dunque quando l'algoritmo trova una sostituzione unificatrice per due termini, tali termini sono unificabili e il termine unificato ottenuto appartiene a $Inst(s) \cap Inst(t)$. Ma l'algoritmo coglie tutti i casi unificabili, ossia è completo?

Proposizione 3 *L'algoritmo della forma risolta è completo.*

Dimostrazione Supponiamo che due termini siano unificabili ma l'algoritmo non trova una sostituzione unificatrice per essi. Poiché termina in ogni caso, l'algoritmo dichiarerebbe fallimento. Tale fallimento non sarebbe corretto, mentre abbiamo visto che ogni caso di fallimento dell'algoritmo risulta corretto. ∎

Proposizione 4 *La sostituzione trovata mediante l'algoritmo della forma risolta è una mgu.*

Dimostrazione Intuitivamente:
l'algoritmo istanzia una variabile soltanto quando questo risulta necessario. Quindi non fa perdere soluzioni mediante una sostituzione che non sia mgu, ossia una sostituzione che fornisca termini meno generali.
Formalmente:
sia θ la sostituzione trovata mediante l'algoritmo. Mostriamo che per ogni altra sostituzione σ, che risulti unificatrice per i due termini s e t, si ha $\theta \leq \sigma$, ossia esiste una sostituzione δ tale che $\delta(\theta(x)) = \sigma(x)$, per ogni x.
Distinguiamo i due casi posibili per una variabile:

1. x è una variabile risolta x_i.
   Dato che una sostituzione viene applicata a tutto E, usiamo la notazione $\sigma(E)$ per indicare che σ viene applicata a tutti i termini di E.
   Banalmente θ unifica E_{ris} in quanto per ogni equazione risolta $x_i = t_i$ si ha, non occorrendo x_i in t_i:
   $$\theta(x_i) = \theta(t_i) = t_i$$
   ossia
   $$\theta(x_i) = t_i$$
   Per ogni altra sostituzione unificatrice σ di E_{ris} si ha
   $$\sigma(\theta(x_i)) = \sigma(t_i) = \sigma(x_i)$$
   ossia
   $$\sigma(\theta(x_i)) = \sigma(x_i)$$
   per cui $\theta \leq \sigma$ relativamente alle variabili risolte.

2. x non è una variabile risolta.
   Per ogni variabile non risolta $x \in V$ si ha d'altra parte $\theta(x) = x$ ed anche quindi
   $$\sigma(\theta(x)) = \sigma(x)$$
   vale a dire $\theta \leq \sigma$.

Concludiamo quindi che $\theta \leq \sigma$ risulta vera per ogni variabile in V. ∎

2.5.3 Antiunificazione

Mentre unificare due termini consiste nel trovare la loro istanza comune piú generale quando questa esiste, antiunificare due termini consiste nel trovare la loro antistanza comune piú particolare, che esiste sempre. Il problema dell'antiunificazione è decidibile.
Un semplice algoritmo per costruire l'antiunificato di due termini in Ter consiste nell'esaminare i due termini in parallelo da sinistra verso destra e, mentre si conservano i simboli corrispondenti uguali, si:
- conserva la variabile se uno dei due simboli corrispondenti è una variabile,
- introduce una nuova variabile se i simboli corrispondenti sono entrambi una costante (con la dovuta attenzione nel dare nomi uguali a variabili nuove che debbono assumere gli stessi valori).

Esempio 4: Siano $a, b, c, f, g, h \in \Sigma$ e $x, y, z \in V$. L'antiunificato dei termini $f(a, b)$ e $g(a, b)$ è x, l'antiunificato di $f(a, x)$ e $f(y, b)$ è $f(y, x)$, e l'antiunificato di $f(x, h(y, a))$ e $f(b, h(c, x))$ è $f(x, h(y, z))$. $\bullet$

3 Calcolo equazionale

Definizione 12 *Siano Σ una segnatura e V un insieme di variabili. Una* Σ-identità *(o* Σ-equazione*) è una coppia $(l, r) \in \mathcal{T}(\Sigma, V) \times \mathcal{T}(\Sigma, V)$, scritta $l = r$. I termini l ed r vengono detti rispettivamente* lato sinistro *e* lato destro *dell'identità $l = r$.*

Nota che molti autori non fanno distinzione tra identità ed equazioni, infatti si dice spesso che un insieme di identità costituisce (la base di) una *teoria equazionale*. In effetti, c'è una differenza logica tra identità ed equazioni. Le identità sono uguaglianze le cui variabili sono implicitamente quantificate universalmente, ovvero uguaglianze che si assumono essere vere indipendentemente da come le variabili vengono istanziate. Le equazioni invece sono uguaglianze che tipicamente vogliamo *risolvere* e le cui variabili sono implicitamente quantificate esistenzialmente (quindi, vogliamo trovare quei valori che, sostituiti alle variabili, rendono vera l'equazione). Nel seguito parleremo indistintamente di identità o equazioni, tenendo comunque presente la distinzione logica tra di esse.

3.1 Deduzione equazionale

Le identità possono essere usate per trasformare i termini in altri termini "equivalenti" rimpiazzando istanze dei lati sinistri con le corrispondenti istanze dei lati destri e viceversa.

Definizione 13 *Sia E un insieme di Σ-identità. La* relazione di riduzione *$\longrightarrow_E \subseteq \mathcal{T}(\Sigma, V) \times \mathcal{T}(\Sigma, V)$ è definita come segue: dati $s, t \in \mathcal{T}(\Sigma, V)$,*

$s \longrightarrow_E t$ se e solo se $\exists\, (l = r) \in E\ \exists p \in Pos(s)\ \exists \sigma$ sostituzione tale che $s|_p = \sigma(l)$ e $t = s[\sigma(r)]_p$.

Esempio 5: Sia data una segnatura $\Sigma = \{e, i, f\}$, dove e è un simbolo di costante, i è un simbolo di funzione unaria ed f è un simbolo di funzione binaria. Siano $x, y, z \in V$ ed E l'insieme con le seguenti identità:

$$\begin{aligned} f(x, f(y, z)) &= f(f(x, y), z) && (1)\\ f(e, x) &= x && (2)\\ f(i(x), x) &= e && (3) \end{aligned}$$

Le identità in E descrivono la teoria dei gruppi e possono essere usate per ridurre i termini alla loro forma normale rispetto a tale teoria. Sia $t = f(i(e), f(e, e))$. Tale termine può essere ridotto alla forma normale e tramite due sequenze diverse di passi di riduzione. Nella prima vengono applicate le identità (1), (3) e (2):

$$f(i(e), f(e, e)) \longrightarrow_E f(f(i(e), e), e) \longrightarrow_E f(e, e) \longrightarrow_E e$$

mentre nella seconda vengono usate la (2) e la (3):

$$f(i(e), f(e, e)) \longrightarrow_E f(i(e), e) \longrightarrow_E e.$$

•

Ricordiamo che la relazione di convertibilità $\overset{*}{\longleftrightarrow}_E$ è la chiusura riflessiva, transitiva e simmetrica di $\longrightarrow_E$ [12]. Nel fare ragionamento equazionale vorremmo poter decidere se due termini sono uguali modulo una teoria definita tramite un insieme di equazioni E, ossia se $s \overset{*}{\longleftrightarrow}_E t$ per $s, t \in \mathcal{T}(\Sigma, V)$. La relazione $\overset{*}{\longleftrightarrow}_E$ può essere caratterizzata sintatticamente nel modo seguente.

Definizione 14 *Sia $\mathcal{R}$ una relazione binaria su $\mathcal{T}(\Sigma, V)$.*
i) $\mathcal{R}$ è chiusa rispetto alle sostituzioni *se e solo se $s\,\mathcal{R}\,t$ implica $\sigma(s)\,\mathcal{R}\,\sigma(t)$ per ogni $s, t \in \mathcal{T}(\Sigma, V)$ e sostituzione σ.*
ii) $\mathcal{R}$ è chiusa rispetto alle operazioni in Σ *se e solo se $s_1\,\mathcal{R}\,t_1, \ldots, s_n\,\mathcal{R}\,t_n$ implica $f(s_1, \ldots, s_n)\,\mathcal{R}\,f(t_1, \ldots, t_n)$ per ogni $n \geq 0$, $f \in \Sigma^n$ ed $s_1, \ldots, s_n, t_1, \ldots, t_n \in \mathcal{T}(\Sigma, V)$.*

Proposizione 5 *Sia E un insieme di Σ-identità. La relazione $\overset{*}{\longleftrightarrow}_E$ è la piú piccola relazione di equivalenza su $\mathcal{T}(\Sigma, V)$ che contiene E e che è chiusa rispetto alle sostituzioni e alle operazioni in Σ.*

Quindi, a partire dall'insieme E di Σ-identità, la relazione $\overset{*}{\longleftrightarrow}_E$ può essere ottenuta chiudendo E rispetto a riflessività, simmetria, transitività, sostituzioni e operazioni in Σ. Questo procedimento di chiusura può essere descritto tramite un sistema di *regole di inferenza* che permettono di fare *deduzione equazionale*, ovvero derivare a partire da E tutte le equazioni $s = t$ che sono *conseguenze sintattiche di E*, denotato mediante $E \vdash s = t$. La seguente è una caratterizzazione alternativa della relazione $\overset{*}{\longleftrightarrow}_E$.

Definizione 15 *Sia E un insieme di* Σ*-identità. Le* conseguenze sintattiche di E *sono derivate tramite le seguenti regole di inferenza:*

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

$$\frac{}{E \vdash t = t} \qquad \frac{E \vdash s = t}{E \vdash t = s} \qquad \frac{E \vdash s = t \quad E \vdash t = u}{E \vdash s = u}$$

$$\frac{E \vdash s = t}{E \vdash \sigma(s) = \sigma(t)} \qquad \frac{E \vdash s_1 = t_1 \quad \ldots \quad E \vdash s_n = t_n}{E \vdash f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)}$$

Proposizione 6 *Sia E un insieme di* Σ*-identità. La relazione* $\overset{*}{\longleftrightarrow}_E$ *generata da E è l'insieme di identità ottenuto applicando le regole del sistema deduttivo di Definizione 15.*

Quindi, per ogni $s, t \in \mathcal{T}(\Sigma, V)$, si ha che $s \overset{*}{\longleftrightarrow}_E t$ se e solo se $E \vdash s = t$ in un numero finito di passi tramite le regole del sistema deduttivo. Tali regole consentono di produrre *alberi di prova* [1].

3.2 Modelli

Finora abbiamo introdotto nozioni a livello sintattico senza darne un'interpretazione semantica ed abbiamo trasformato i termini manipolando la loro rappresentazione simbolica. Data una segnatura Σ, un Σ-modello fornisce un'interpretazione dei simboli di funzione in Σ.

Definizione 16 *Sia* Σ *una segnatura. Un* Σ-modello *(o semplicemente* modello*) è una coppia* (A, F)*, in cui A è un insieme ed F è una famiglia di funzioni,* $F = \{F_f \mid f \in \Sigma\}$*, tale che*
i) $F_f \in A$ *per ogni* $f \in \Sigma^0$
ii) $F_f \in A \times \ldots \times A \to A$ *per ogni* $f \in \Sigma^n$ *con* $n > 0$.

Esempio 6: Con riferimento alla segnatura Σ dell'Esempio 1, una possibile interpretazione semantica dei simboli di funzione in Σ è il Σ-modello (A, F), in cui $A = \mathbb{N}$, $F_0 = \underline{0} \in \mathbb{N}$, F_{succ} è la funzione successore su $\mathbb{N}$ ed $F_=$ è la funzione uguaglianza su $\mathbb{N}$. •

Data una segnatura Σ, indichiamo la classe di tutti i Σ-modelli con Σ-mod. Nel seguito indicheremo un Σ-modello (A, F) solo mediante A.

Siano $A \in \Sigma$-mod ed $\mathcal{R}$ una famiglia di relazioni tale che $\mathcal{R}$ è una relazione di equivalenza su A. Se tutte le funzioni f_A di A applicate ad elementi equivalenti producono elementi equivalenti, allora $\mathcal{R}$ è detta chiusa rispetto alle operazioni di A.

Definizione 17 *$\mathcal{R}$ si dice* chiusa rispetto alle operazioni in *$A \in \Sigma$-mod se e solo se per ogni $f \in \Sigma$ e $a_i, b_i \in A$ $(i = 1, \ldots, n)$ si ha che*
$a_1 \,\mathcal{R}\, b_1, \ \ldots, \ a_n \,\mathcal{R}\, b_n$ implica $f_A(a_1, \ldots, a_n) \,\mathcal{R}\, f_A(b_1, \ldots, b_n)$.

Data una segnatura Σ ed un insieme V di variabili, il modello dei termini (*modello sintattico*) è definito prendendo come universo lo stesso insieme $\mathcal{T}(\Sigma, V)$ e come interpretazione dei simboli di funzioni i simboli stessi.

Definizione 18 *Siano Σ una segnatura e V un insieme di variabili.*
Il modello dei termini *$\mathcal{T}(\Sigma, V)$ è la coppia*

$$(\mathcal{T}(\Sigma, V) \ , \ \{f_{\mathcal{T}} \mid f \in \Sigma\})$$

dove
i) $f_{\mathcal{T}} = f$ per ogni $f \in \Sigma^0$
ii) $f_{\mathcal{T}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, V)$ per ogni $f \in \Sigma^n$ con $n > 0$ e $t_i \in \mathcal{T}(\Sigma, V)$ $(i = 1, \ldots, n)$.

La Definizione 15 ha caratterizzato la relazione $\overset{*}{\longleftrightarrow}_E$ come l'insieme di tutte le identità che possono essere dedotte sintatticamente da E mediante le regole di inferenza del sistema deduttivo. Una caratterizzazione semantica di $\overset{*}{\longleftrightarrow}_E$ si basa invece sulla nozione di *validità* di un'identità in un Σ-modello A, nel senso che una identità $s = t$ è vera se s e t denotano lo stesso elemento di A per ogni valutazione delle variabili di s e t.

Definizione 19 *Siano $A \in \Sigma$-mod ed $s = t$ una Σ-identità (o Σ-equazione). Diciamo che $s = t$ è* valida *in A o che A è un* modello *di $s = t$, denotato mediante $A \models s = t$, se e solo se per ogni assegnamento $\nu : V \to A$ si ha*

$\nu(s) = \nu(t)$. [1]
Sia E un insieme di equazioni. A è un modello *di E, denotato da $A \models E$, se e solo se $A \models s = t$ per ogni $(s = t) \in E$.*
Un'equazione $s = t$ è soddisfatta in *A se e solo se esiste un assegnamento $\nu : V \to A$ tale che $\nu(s) = \nu(t)$.*

Definizione 20 *Sia E un insieme di equazioni. La* varietà *definita da E è l'insieme $\mathcal{M}(E)$ di tutti i modelli di E:*

$$\mathcal{M}(E) = \{A \in \Sigma\text{–}mod \mid A \models E\}.$$

Definizione 21 *Sia E un insieme di equazioni. L'identità $s = t$ è una* conseguenza semantica *di E (denotato tramite $E \models s = t$) se e solo se è valida in tutti i modelli di E, ovvero per ogni $A \in \mathcal{M}(E)$ si ha che $s = t$ è valida in A.*
La relazione $=_E = \{(s,t) \in \mathcal{T}(\Sigma, V) \times \mathcal{T}(\Sigma, V) \mid E \models s = t\}$ è la teoria equazionale *indotta da E, mentre E è la* base *o l'*insieme degli assiomi della teoria.

Nel seguito, quando considereremo una teoria equazionale, sarà sufficiente evidenziare gli assiomi E della teoria e parleremo (anche se impropriamente) di teoria equazionale E.

Ogni modello appartenente alla varietà di un insieme di equazioni E soddisfa non solo le equazioni in E, ma anche tutte le equazioni che possono essere dedotte equazionalmente da quelle in E. La relazione $\stackrel{*}{\longleftrightarrow}_E$ è la minima relazione di equivalenza su $\mathcal{T}(\Sigma, V)$ chiusa rispetto alle operazioni ed alle sostituzioni. Tale teoria è costituita da tutte le equazioni valide nella varietà di E: a partire da E ed applicando le regole di inferenza del sistema deduttivo di Definizione 15, si possono dedurre tutte le equazioni vere nella teoria. Quanto detto si riassume nel noto **teorema di completezza di Birkhoff** che afferma che la relazione $=_E$ definita in modo semantico (*model-teoretico*) coincide con la relazione $\stackrel{*}{\longleftrightarrow}_E$ definita in modo sintattico (*proof-teoretico*).

Teorema 1 *Siano E un insieme di equazioni ed $s = t$ un'equazione. Allora si ha che $s =_E t$ se e solo se $E \vdash s = t$.*

Osservazione 4: Per un trattamento piú generale sui modelli di teorie equazionali tramite *algebre* e *semantica algebrica*, rimandiamo a [5]. ∎

[1] Ricordiamo che una funzione da V in elementi di un modello A può essere sempre estesa in modo univoco ad un omomorfismo da $\mathcal{T}(\Sigma, V)$ in A.

4 Sistemi di riscrittura

Un sistema di riscrittura di termini è un particolare sistema di riduzione astratto $\langle A, \longrightarrow \rangle$, dove A è un insieme di termini e $\longrightarrow$ è una relazione di riduzione, detta anche relazione di riscrittura. Vari sistemi algoritmici noti, quali le macchine di Turing, il λ-calcolo, etc. possono essere visti come sistemi di riscrittura. I sistemi che noi tratteremo nel seguito sono sistemi del prim'ordine, dove l'insieme dei termini è $\mathcal{T}(\Sigma, V)$ e la relazione di riscrittura è definita tramite un insieme di regole di riscrittura su $\mathcal{T}(\Sigma, V)$.

Ricordiamo che la semantica di un linguaggio di programmazione funzionale, quale ML, SML, CAML, etc. è data essenzialmente tramite un sistema di riscrittura. La definizione di una funzione in un linguaggio funzionale non è altro che un sistema di riscrittura, ovvero un insieme di regole che asseriscono come valutare l'applicazione di tale funzione a determinati argomenti. La valutazione di un'espressione del linguaggio consiste in un processo di trasformazione (riscrittura) dell'espressione data fino a quando non è possibile svolgere ulteriori trasformazioni. Il risultato di tale processo è il valore dell'espressione, ovvero la sua forma normale rispetto alle regole di trasformazione.

Il processo di trasformazione in forma normale può essere svolto applicando diverse strategie di valutazione o di riduzione (Sezione 10). Una strategia definisce come scegliere, ad ogni passo, quale sottoespressione ridurre per prima fra tutte quelle riducibili (redessi). Si hanno, ad esempio, la strategia di valutazione "call-by-value" (tipica dei linguaggi della famiglia ML) e la strategia di valutazione "lazy evaluation" (tipica dei linguaggi come Miranda e Haskell).

Definizione 22 *Un* sistema di riscrittura di termini *(abbreviato in* srt *o* trs *dall'inglese* term rewriting system*) R su una segnatura Σ è un insieme $\{(l_i, r_i) \mid l_i, r_i \in \mathcal{T}(\Sigma, V),\ l_i \notin V,\ Var(r_i) \subseteq Var(l_i)\}$. Le coppie (l_i, r_i) sono dette* regole di riscrittura *e si scrivono $l_i \longrightarrow r_i$ (equazioni orientate).*
La relazione di riscrittura *$\longrightarrow_R$ su $\mathcal{T}(\Sigma, V)$ è definita come la piú piccola relazione contenente R e chiusa rispetto alle sostituzioni e alle operazioni in Σ (o contesti):*

i) $t \longrightarrow_R s$ *implica* $\sigma(t) \longrightarrow_R \sigma(s)$ *per ogni sostituzione* σ;
ii) $t \longrightarrow_R s$ *implica* $f(\ldots, t, \ldots) \longrightarrow_R f(\ldots, s, \ldots)$ *per ogni* $f \in \Sigma^n, n>0$.

4.1 Redesso, derivazione, forma normale

Definizione 23 *Dato un srt R su Σ, un termine t* si riscrive *(o* si riduce*) in un termine s, denotato da $t \longrightarrow_R s$, se esiste una regola $l \longrightarrow r$ in R, una sostituzione σ ed una occorrenza $t|_p$ (detta* redesso*) alla posizione p, tale che $t|_p = \sigma(l)$ ed $s = t[\sigma(r)]_p$.*

Esempio 7: Sia R il seguente srt:

$$\begin{array}{rcl} f(h(x)) & \longrightarrow & h(x) \\ h(h(x)) & \longrightarrow & x \\ h(a) & \longrightarrow & a \end{array}$$

Sia dato il termine $t = h(f(h(b)))$ e vediamo quali riduzioni possiamo applicare a partire da t in R. Le posizioni di t sono $Pos(t) = \{\epsilon, 1, 1.1, 1.1.1\}$. L'unica riduzione possibile è quella con la prima regola con $p = 1$, $t|_p = f(h(b))$ e $\sigma = \{b/x\}$, da cui si ha $t \longrightarrow_R t' = t[\sigma(h(x))]_p = h(h(b))$. Dato $Pos(t') = \{\epsilon, 1, 1.1\}$, si ha un altro passo di riduzione con la seconda regola e $p = \epsilon$, $t'|_p = t'$ e $\sigma' = \{b/x\}$: $t' \longrightarrow_R t'' = t'[\sigma'(x)]_p = b$. Quindi la sequenza di riduzione è la seguente: $h(f(h(b))) \longrightarrow_R h(h(b)) \longrightarrow_R b$. •

D'ora in avanti scriveremo $\longrightarrow$ al posto di $\longrightarrow_R$, quando R sia noto dal contesto. Ricordiamo che un termine t è *in forma normale* rispetto ad un srt R se e solo se non esiste alcun termine s tale che $t \longrightarrow s$, ovvero nessun sottotermine di t è un'istanza del lato sinistro di una qualche regola di R. Inoltre, un termine s è *una forma normale* del termine t rispetto ad R se e solo se $t \stackrel{*}{\longrightarrow} s$ ed s è in forma normale rispetto ad R. Richiamiamo la seguente notazione che useremo: dati due termini s, t ed una relazione $\longrightarrow$, la notazione $s\downarrow t$ indica che s e t hanno un successore in comune, ovvero esiste un termine u tale che $s \stackrel{*}{\longrightarrow} u \stackrel{*}{\longleftarrow} t$. Una forma normale di un termine t rispetto ad un srt R sarà indicata con $t\downarrow$.

5 Sistemi di riscrittura terminanti

Sappiamo [12] che la terminazione (SN) di un sistema di riscrittura assicura l'esistenza della forma normale per un qualsiasi termine, in quanto non esistono sequenze con un numero infinito di passi di riscrittura. Purtroppo il problema della terminazione di un sistema di riscrittura è, in generale, indecidibile. Ciò vale anche se le regole contengono solo simboli di funzione

unarie, o se si considera un sistema con una sola regola di riscrittura contenente però simboli di funzioni di arità maggiore di 1.
Infatti ogni macchina di Turing può essere rappresentata come un sistema di riscrittura in modo tale che ad ogni computazione della macchina di Turing corrisponde una riduzione del sistema di riscrittura corrispondente: quindi la decidibilità della terminazione nei sistemi di riscrittura renderebbe decidibile il *problema della fermata (Halting Problem)*.
Il problema della terminazione diventa decidibile nel caso di sistemi di riscrittura chiusi, ovvero le cui regole non contengono variabili, come vedremo piú avanti in questo capitolo dopo aver trattato alcuni dei possibili approcci per provare la terminazione.
Il problema della terminazione si incontra in molti contesti piú o meno astratti in Informatica e in Matematica. Per decidere la terminazione risulta comunemente utile usare:

- ordinamenti parziali dei quali sono note proprietà utili allo scopo;
- immersione (*embedding*) di un sistema in un altro terminante già noto.

L'indecidibilità, in generale, della terminazione di un sistema di riscrittura (finito) costringe a cercare delle condizioni sufficienti che permettano di stabilire se un sistema di riscrittura è terminante. L'idea è basata sull'intuizione di cosa effettua un sistema di riscrittura e cioé che riducendo un termine attraverso una relazione di riscrittura, la complessità strutturale del termine ridotto sia in un certo senso minore di quella del termine di partenza. Evidentemente tale intuizione mal si concilia con la nozione di riscrittura infinita e perciò si lega al concetto di terminazione. L'obiettivo è quindi quello di stabilire un ordinamento (parziale) sull'insieme $\mathcal{T}(\Sigma, V)$ dei termini che permetta di provare che se $t \longrightarrow s$ allora risulti $t \succ s$ nell'ordinamento $\succ$ stabilito su $\mathcal{T}(\Sigma, V)$. Quindi basterebbe trovare un ordinamento benfondato $\succ$ sui termini tale che $t \longrightarrow s$ implichi $t \succ s$ per tutti i termini t, s. Ciò vuol dire che un sistema di riscrittura R è terminante se e solo se la relazione di riscrittura $\longrightarrow_R$ è contenuta nella relazione $\succ$.

Questo risultato però non ci aiuta molto poiché esprime una condizione su tutte le possibili riduzioni. Si vuole invece tentare di stabilire un risultato analogo che tenga conto della struttura dei termini e del modo in cui questi possono ridursi. Una relazione di riscrittura è, per definizione, chiusa rispetto ai contesti e alle sostituzioni. L'obiettivo è quello di definire degli ordinamenti sui termini che siano benfondati e chiusi rispetto a contesti e sostituzioni, in modo tale che, una volta assunto un ordinamento con tali proprietà, per determinare se un sistema di riscrittura è terminante, è sufficiente verificare una condizione sulle regole del sistema di riscrittura (che sono in numero finito) e non piú su tutte le possibili riduzioni.

5.1 Ordinamenti parziali utili per la terminazione

Consideriamo gli ordinamenti di riduzione ed alcuni loro casi particolari.

5.1.1 Ordinamenti di riduzione

La relazione $\succ \subset Ter^2$ è un *ordinamento di riduzione* se è:

- parziale;
- stretto (irriflessivo e transitivo);
- antisimmetrico;
- WF o noetheriano;
- chiuso per contesti (detto anche *monotono* o *compatibile con le operazioni*):
$$t \succ t' \Longrightarrow C[t] \succ C[t']$$
per ogni contesto $C[]$, ovvero
$$t \succ t' \Longrightarrow f(\ldots, t, \ldots) \succ f(\ldots, t', \ldots)$$
per ogni $f \in \Sigma$;
- chiuso per sostituzioni (detto anche *stabile*):
$$t \succ t' \Longrightarrow \sigma(t) \succ \sigma(t')$$
per ogni sostituzione σ.

Esempio 8: Sia $t \succ t'$ se $|t| > |t'|$, dove $|.| : Ter \longrightarrow \mathbb{N}$, con $|t| \geq 1$, dà il numero di caratteri che compongono t considerato come una stringa.
Verificare che $\succ$ non è un ordinamento di riduzione mancando la chiusura per sostituzioni. •

Esempio 9: Sia $t \succ t'$ se $|t| > |t'|$ e $|t|_x > |t'|_x$ per ogni variabile x, dove $|t|_x$ calcola soltanto la lunghezza rispetto alle occorrenze in t della variabile x.
Verificare che $\succ$ è un ordinamento di riduzione. •

Esempio 10: Sia $t \succ t'$ se $|t| > |t'|$ oppure $|t| = |t'|$ e $t >_{lex} t'$.
Verificare che $\succ$ non è un ordinamento di riduzione mancando la chiusura per sostituzioni (se $f >_{lex} g$ ma non piú, ad esempio, $f(x)$ di $g(g(x)) = \{g(x)/x\}(g(x))$). •

Proposizione 7 *(Teorema di Lankford) Un sistema di riscrittura R è SN se e solo se esiste un ordinamento di riduzione $\succ$ tale che $l \succ r$ per ogni $l \longrightarrow r$ di R.*

Dimostrazione Se $l \succ r$ per ogni $l \longrightarrow r$ di R, allora $\longrightarrow_R \subseteq \succ$ ed anche $\overset{*}{\longrightarrow}_R \subseteq \succ$, essendo $\succ$ transitivo. Inoltre essendo $\succ$ WF, ogni riduzione è finita in quanto la relazione $\longrightarrow_R$ è indotta dalle regole.
Viceversa, se R è SN, possiamo definire $t \succ t'$ se $t \overset{+}{\longrightarrow}_R t'$.
Poiché $\longrightarrow_R \subseteq \overset{+}{\longrightarrow}_R$, si ha $l \succ r$ se $l \longrightarrow r$ per ogni regola di R.
L'ordinamento cosí definito risulta essere di riduzione in quanto dall'essere R SN segue che $\succ$ è irriflessivo, stretto, transitivo, antisimmetrico, WF e chiuso per sostituzioni e contesti. ■

Particolari ordinamenti di riduzione sono gli ordinamenti di semplificazione.

5.1.2 Ordinamenti di semplificazione

La relazione $\succ \subset Ter^2$ è un *ordinamento di semplificazione* se è:

- parziale;
- stretto;
- antisimmetrico;
- ha la *proprietà del sottotermine* (proprietà piú forte della WF), ossia:

$$t = C[t'] \succ t'$$

per ogni contesto $C[] \neq []$ di t'. Ovvero, usando le posizioni,

$$t \succ t|_p$$

per ogni posizione $p \in Pos(t) \setminus \{\epsilon\}$;

- chiuso per contesti;
- chiuso per sostituzioni.

Risulta facile definire degli ordinamenti di semplificazione, dato che ogni t' sottotermine proprio di t è tale che $t \succ t'$.

Esempio 11: Verificare che $>_{lex}$ non è semplificante mancando la proprietà del sottotermine (se $f >_{lex} g$ allora $f(g(f(x))) >_{lex} g(f(x))$, ma $g(f(x))$ non lo è di $f(x)$). •

Ogni ordinamento di semplificazione su una segnatura finita contiene l'ordinamento di *sprofondamento* (per una definizione rigorosa di immersione omeomorfa vedasi [1]).

Possiamo grosso modo dire che il termine s è *immerso* o *sprofondato* nel termine t se s può essere ottenuto da t cancellando 'opportunamente' alcuni nodi e ricomponendo. Ad esempio, $f(f(x))$ è sprofondato in $g(f(f(x)))$ ma anche in $f(g(f(x)))$.

Esiste il seguente noto teorema di Kruskal sugli alberi etichettati su un alfabeto finito.

Teorema 2 *(Teorema di Kruskal) Ogni successione infinita senza cicli di alberi etichettati su un alfabeto finito è tale che esiste un albero t_i sprofondato in un t_j, con $j > i$.*

Per la dimostrazione vedasi [1].

Teorema 3 *Ogni ordinamento di semplificazione sulle regole di riscrittura garantisce la terminazione.*

Dimostrazione Per assurdo, supponiamo che $l \succ r$ per ogni regola di R, con $\succ$ ordinamento di semplificazione, ma che esista una riduzione infinita

$$t_1 \longrightarrow t_2 \ldots \longrightarrow t_i \ldots \longrightarrow t_j \ldots$$

dove si ha

$$t_1 \succ t_2 \ldots \succ t_i \ldots \succ t_j \ldots$$

essendo tale riduzione effettuata mediante le regole di R. Per il teorema di Kruskal esiste una coppia (t_{i_k}, t_{j_k}), con $j_k > i_k$ per cui $t_{i_k} \succ t_{j_k}$, con t_{i_k} sprofondato in t_{j_k}. Essendo $\succ$ un ordinamento di semplificazione, deve aversi $t_{j_k} \succ t_{i_k}$, da cui la contraddizione in assenza di cicli. ■

Un modo alternativo per ottenere il risultato enunciato dal teorema precedente è dato dal seguente teorema, insieme alla Proposizione 7:

Teorema 4 *Sia Σ una segnatura finita. Ogni ordinamento di semplificazione $\succ$ su $\mathcal{T}(\Sigma, V)$ è un ordinamento di riduzione.*

Per la dimostrazione vedasi [1], dove viene utilizzato l'ordinamento di immersione (embedding).

Particolari ordinamenti di riduzione sono gli ordinamenti per cammino ricorsivo (*rpo*, dall'inglese *recursive path ordering*).

5.1.3 Ordinamenti per cammino ricorsivo

Dobbiamo premettere l'estensione di ordinamenti a multinsiemi finiti (il cui tipo di dato assumiamo di conoscere).
Sia dato $(A, \succ)$, A insieme (nel nostro caso Ter). L'ordinamento $\succ$ si estende all'ordinamento $\succ\!\succ$ sui multinsiemi finiti su A, ottenendo $(MA, \succ\!\succ)$, definendo, con M metavariabile per multinsieme:

Definizione 24 *$M_1 \succ\!\succ M_2$ se $M_1 \neq M_2$ e per ogni $m_2 \in M_2 \backslash M_1$ esiste $m_1 \in M_1 \backslash M_2$ tale che $m_1 \succ m_2$.*

Esempio 12: $M_1 = \{3, 3, 4, 0\} \succ\!\succ M_2 = \{3, 2, 2, 1, 1, 1, 4\}$ in quanto
$M_2 \backslash M_1 = \{2, 2, 1, 1, 1\}$
$M_1 \backslash M_2 = \{3, 0\}$
e si ha $2 \prec 3$ e $1 \prec 3$.
Mentre la relazione non sussiste tra $M_1 = \{3, 1, 1\}$ e $M_2 = \{4, 1\}$ in quanto
$M_2 \backslash M_1 = \{4\}$
$M_1 \backslash M_2 = \{3, 1\}$
e si ha che né $3 \succ 4$ né $1 \succ 4$. •

Proposizione 8 *Se $\succ$ è WF allora anche $\succ\!\succ$ lo è.*

Dimostrazione Per assurdo, supponendo che $\succ\succ$ non sia WF, esisterebbe una catena infinita

$$M_1 \succ\succ M_2 \succ\succ \ldots \succ\succ M_i \succ\succ \ldots$$

Ciò equivale ad ottenere un albero infinito a ramificazione finita rispetto a $\succ\succ$ con radice in M_1. Esiste quindi un albero infinito a ramificazione finita rispetto a $\succ$ con radice in $t_1 \in M_1$, per qualche $t_1 \in M_1$. Per il lemma di König, esiste un cammino infinito rispetto a $\succ$, in contraddizione con l'ipotesi di benfondatezza di $\succ$. ∎

L'ordinamento $\succ\succ$ può essere definito come la chiusura transitiva del rimpiazzamento di un arbitrario elemento di un multinsieme con un numero finito, quindi anche nullo, di elementi a lui minori rispetto a $\succ$.

Possiamo ora definire l' ordinamento *rpo* per termini chiusi, che poi estenderemo ai termini aperti. Tale ordinamento è parametrico rispetto ad un ordinamento $>$, definito sulla segnatura Σ, che fissa le precedenze tra gli operatori.

Definizione 25 *Sia dato* $(\Sigma, >)$ *con* $>$ *ordinamento parziale su* Σ. *L'ordinamento* rpo *su* $\mathcal{T}(\Sigma)$ *è definito ricorsivamente come segue.*

$$s = f(s_1, \ldots, s_m) \succ_{rpo} t = g(t_1, \ldots, t_n)$$

se:

i) $f = g$ *e* $\{s_1, \ldots, s_m\} \succ\succ_{rpo} \{t_1, \ldots, t_n\}$

ii) $f > g$ *e* $\{s\} \succ\succ_{rpo} \{t_1, \ldots, t_n\}$

iii) $f \not\geq g$ *e* $\{s_1, \ldots, s_m\} \succ\succeq_{rpo} \{t\}$

dove $\succ\succeq_{rpo}$ *è* $\succ\succ_{rpo}$ *oppure l'uguaglianza* $\sim$, *tenendo conto che due termini sono considerati uguali se lo sono a meno di permutazioni degli argomenti.*

Osserviamo la forzatura sui termini $f(t_1, t_2) = f(t_2, t_1)$ implicita nella definizione appena data.

L'ordinamento *rpo* viene esteso ai termini aperti aggiungendo semplicemente ai casi della Definizione 25, un caso ulteriore che tratta l'introduzione di variabili in modo tale che l'ordinamento risulti stabile.

Definizione 26 *L'ordinamento* rpo generalizzato *su* $\mathcal{T}(\Sigma, V)$ *è definito come segue: dati due termini* s *e* t*, si ha che* $s \succ_{rpo} t$ *se valgono i), ii) o iii) di Definizione 25 oppure*
iv) $s \notin V$ *e* $t \in Var(s)$.

Teorema 5 *Ogni rpo è un ordinamento di semplificazione.*

Dimostrazione Vedasi [1]. ∎

Esempio 13: Sia R il seguente sistema sulla segnatura $\Sigma = \{a, b, c, f, g, h\}$:

$$\begin{array}{rcl} f(a,b) & \longrightarrow & g(c) \\ h(f(a,c)) & \longrightarrow & h(b) \\ c & \longrightarrow & b \end{array}$$

Se prendiamo un ordinamento *rpo* basato sulle precedenze $h > g > f > a > b > c$, si può verificare che in $f(a,b) \longrightarrow g(c)$ non si ha $f(a,b) \succ_{rpo} g(c)$ e in $c \longrightarrow b$ non si ha $c \succ_{rpo} b$. •

Esempio 14: Sia R il seguente sistema sulla segnatura $\Sigma = \{a, f, g, h\}$:

$$\begin{array}{rcl} f(x,a) & \longrightarrow & g(x) \\ f(x,g(y)) & \longrightarrow & g(f(x,y)) \\ h(x,a) & \longrightarrow & x \\ h(x,g(y)) & \longrightarrow & f(h(x,y),x) \end{array}$$

Si vuole determinare un ordinamento di riduzione sui termini tale che R sia terminante rispetto a tale ordinamento. Basta prendere un ordinamento *rpo* basato ad esempio sulle precedenze $h > f > g$.
Infatti, si può verificare che per ogni regola $l \longrightarrow r$ di R si ha $l \succ_{rpo} r$. •

5.2 Casi di terminazione decidibile

Premettiamo alcune definizioni.

Definizione 27 *Un sistema di riscrittura è detto:*

lineare *se ogni variabile ha al piú una occorrenza. Quindi un sistema di riscrittura chiuso, ossia con ogni sua regola* $l \longrightarrow r$ *avente* l *ed* r *chiusi, è lineare.*

lineare a sinistra *se in ogni sua regola* $l \longrightarrow r$ *si ha che* l *è lineare.*

lineare a destra *se in ogni sua regola* $l \longrightarrow r$ *si ha che* r *è lineare. Quindi* chiuso a destra *è caso particolare di lineare a destra.*

ortogonale *se è lineare a sinistra e non ambiguo (vedi Definizione 31).*

Esempio 15: $f(x, g(y)) \longrightarrow g(f(g(y), x))$ è una regola lineare; $f(x, x) \longrightarrow f(a, b)$ è lineare a destra, in particolare chiusa a destra; $f(x) \longrightarrow g(x, x)$ è lineare a sinistra.
$R = \{and(x, true) \longrightarrow x, \;\; and(x, false) \longrightarrow false\}$ è ortogonale;
$R = \{or(x, y) \longrightarrow x, \;\; or(x, y) \longrightarrow y\}$ non lo è essendo ambiguo;
$R = \{or(x, true) \longrightarrow true, \;\; or(x, false) \longrightarrow x\}$ è ortogonale. •

Si vuole decidere la terminazione osservando soltanto alcune particolari riduzioni, in particolare quelle che partono dai lati sinistri delle regole e che sono dette *catene*.

Definizione 28 *Dato un insieme finito di regole di riscrittura* $\{l_i \longrightarrow r_i\}$, *una* catena *è:*

1. *ogni riduzione che inizia con* l_i*;*
2. *se si ha* $l_i \longrightarrow \ldots \longrightarrow C_i[s]$ *con* $s \notin V$ *e unificabile con* l_i *via l'mgu* θ*, ogni riduzione che inizia con* $\theta(l_i) = C_i[\theta(s)]$*.*

Esempio 16: La regola $f(x, g(y)) \longrightarrow g(f(g(y), x))$ è una catena dalla quale origina la catena infinita

$$g(f(g(y), g(y))) \longrightarrow g(g(f(g(y), g(y)))) \longrightarrow \ldots g^n(f(g(y), g(y))) \longrightarrow \ldots$$

La regola $f(f(x)) \longrightarrow f(x)$ è una catena dalla quale originano un numero infinito di catene finite.
Mediante l'mgu $\theta = \{f(x)/x\}$ si ha

$$f^3(x) \longrightarrow f^2(x) \longrightarrow f(x)$$

Mediante l'mgu $\theta = \{f^2(x)/x\}$ tra $f(x)$ e $f(f(x))$ si ha

$$f^4(x) \longrightarrow f^3(x) \longrightarrow f^2(x) \longrightarrow f(x)$$

e cosí via, tutte con diverso termine iniziale. •

Si hanno i seguenti risultati:

Teorema 6 *Un sistema di riscrittura ortogonale oppure lineare a destra è SN se e solo se non ha catene infinite.*

Teorema 7 *Un sistema di riscrittura ortogonale oppure lineare a destra non è SN se e solo se ha qualche catena che cicla oppure ha infinite catene finite.*

Corollario 1 *Se un sistema di riscrittura è ortogonale oppure lineare a destra, la terminazione è decidibile se ha un numero finito di catene con differenti termini iniziali.*

Dimostrazione Si generano tutte le catene e si verifica quanto detto. ∎

Corollario 2 *La terminazione di un sistema di riscrittura chiuso a destra è decidibile.*

Dimostrazione Si generano tutte le catene che in questo caso iniziano con i lati sinistri delle regole non istanziati. Quindi vi sono al più tante catene con termini iniziali differenti quante sono le regole. Basta quindi verificare se ciclano. ∎

Esempio 17: Dalle regole $f(x,x) \longrightarrow f(a,b)$ e $b \longrightarrow c$ si hanno le catene

$$f(x,x) \longrightarrow f(a,b) \longrightarrow f(a,c)$$

$$b \longrightarrow c$$

per cui il sistema di riscrittura è SN.
Aggiungendo la regola $a \longrightarrow c$ il sistema diviene ambiguo e si perde SN. Infatti si ha la catena infinita

$$f(x,x) \longrightarrow f(a,b) \longrightarrow f(a,c) \longrightarrow f(c,c) \longrightarrow f(a,b) \longrightarrow \dots$$

Aggiungendo la regola $b \longrightarrow a$ il sistema diviene ambiguo e si perde SN. Infatti si ha la catena infinita

$$f(x,x) \longrightarrow f(a,b) \longrightarrow f(a,a) \longrightarrow f(a,b) \longrightarrow \dots$$

•

Un controesempio per il caso non lineare a destra è il seguente:

Esempio 18: Dalle regole $f(a,b,x) \longrightarrow f(x,x,b)$ e $b \longrightarrow a$ si hanno le catene

$$f(a,b,x) \longrightarrow f(x,x,b) \longrightarrow f(x,x,a)$$
$$b \longrightarrow a$$

Ma il criterio non funziona. Infatti il termine $f(a,b,b)$ cicla:

$$f(a,b,b) \longrightarrow f(b,b,b) \longrightarrow f(a,b,b) \longrightarrow \dots$$

•

6 Sistemi di riscrittura confluenti

Sappiamo, dal lemma di Newman visto negli ARS [12], che $SN + WCR \Longrightarrow CR$. Sappiamo, sempre dagli ARS, che tale problema è indecidibile in generale, ma risulta essere decidibile per sistemi di riscrittura terminanti. Infatti, dalle proprietà di terminazione e di Church-Rosser segue il seguente risultato di decidibilità:

Lemma 1 *Se la relazione $\longrightarrow$ è noetheriana e Church-Rosser, allora $\stackrel{*}{\longleftrightarrow}$ è decidibile.*

Dimostrazione Infatti, $s \stackrel{*}{\longleftrightarrow} t$ se e solo se $s\downarrow \equiv t\downarrow$, esistendo uniche le forme normali di ogni termine per ogni sistema di riscrittura canonico. ■

In presenza di sistemi di riscrittura terminanti la confluenza debole garantendo la confluenza, la struttura dei termini permette di ottenere una condizione sufficiente per la confluenza debole, per enunciare la quale ci occorre la nozione di coppia critica.

6.1 Spectrum delle regole di un sistema di riscrittura

Le situazioni nelle sequenze di riduzione che possono compromettere la confluenza debole sono date da quei casi in cui un termine si può riscrivere in piú modi diversi. Quindi, tali situazioni possono essere rilevate andando a considerare quei casi in cui piú di una regola è applicabile allo stesso termine. Poiché una sequenza di riduzione è ottenuta, a partire dalle regole iniziali, per sostituzioni e rimpiazzamenti, possiamo considerare l'insieme di tutte le possibili istanze dei lati sinistri delle regole di R, insieme dato dalla nozione di *spectrum* delle regole [2].

Definizione 29 *Sia dato un sistema di riscrittura R. Lo* spectrum *delle regole di R è definito nel modo seguente:*

$$spectrum(R) = \{\sigma s \mid \exists t.\ (s,t) \in R,\ \sigma \ sostituzione\}$$

Consideriamo ora il seguente insieme B:

$$B = \{(s,t) \mid \exists u, s', t'.\ u \in spectrum(R), s' \longleftarrow u \longrightarrow t',\ s = s'{\downarrow},\ t = t'{\downarrow}\}.$$

Si noti che la definizione data di B non è generale poiché non viene considerato il caso di applicazione di regole a sottotermini. La semplificazione è fatta allo scopo di rendere piú intuitiva ed immediata l'idea delle situazioni critiche da considerare. Rappresentando graficamente le situazioni catturate dalla definizione di B, si ha che guardare agli spectra delle regole di riscrittura significa esaminare le seguenti regole:

$$\begin{array}{rcl} l & \longrightarrow & r \\ \sigma_1 l & \longrightarrow & \sigma_1 r \\ \sigma_2 l & \longrightarrow & \sigma_2 r \\ \ldots & & \end{array}$$

ed individuare quei termini u che compaiono come lati sinistri uguali $\sigma_i l = \tau_j l'$ in due spectra diversi:

$$\begin{array}{lcr} l \longrightarrow r & & l' \longrightarrow r' \\ \sigma_1 l \longrightarrow \sigma_1 r & & \tau_1 l' \longrightarrow \tau_1 r' \\ \sigma_2 l \longrightarrow \sigma_2 r & & \tau_2 l' \longrightarrow \tau_2 r' \\ \ldots & & \ldots \\ & \sigma_i l \longrightarrow \sigma_i r & \\ & \tau_j l' \longrightarrow \tau_j r' & \\ & \ldots & \\ \ldots & & \ldots \end{array}$$

Estendendo il concetto alla situazione piú generale, occorre considerare tutti i casi in cui le istanze di due lati sinistri di regole sono tali che un sottotermine dell'una coincide con tutta l'altra istanza, per cui la prima istanza è un termine che per definizione si riscrive in due modi diversi, con una regola applicata a tutto il termine e l'altra regola applicata sul sottotermine. Queste situazioni sono catturate dalla nozione di coppia critica.

6.2 Coppie critiche

Iniziamo con degli esempi di coppie critiche (cc) ottenute da regole di riscrittura.

Esempio 19:

$$\begin{array}{rcl} 0 + succ(x) & \longrightarrow & succ(x) \\ y + succ(z) & \longrightarrow & succ(y+z) \end{array}$$

Mediante la sostituzione unificatrice mgu $\sigma = \{0/y, x/z\}$ si ha che $\sigma(0 + succ(x)) = \sigma(y + succ(z)) = 0 + succ(x)$. Applicando la prima regola al termine unificato si ottiene $succ(x)$, mentre applicandogli la seconda regola si ottiene $succ(0 + x)$, entrambi in forma normale rispetto alle regole date. Si è cosí ottenuta la cc ($succ(x)$, $succ(0 + x)$). •

Esempio 20:

$$\begin{array}{rl} or(x,y) & \longrightarrow x \\ or(x,y) & \longrightarrow y \end{array}$$

Con la sostituzione vuota id si ha che $or(x, y)$ si riduce mediante la prima regola ad x, mentre mediante la seconda regola si riduce ad y, entrambi in forma normale rispetto alle regole date. Si è cosí ottenuta la cc (x, y). •

Esempio 21:

$$\begin{array}{rcl} f(g(x)) & \longrightarrow & b \\ g(h(a)) & \longrightarrow & c \end{array}$$

Con la sostituzione mgu $\theta = \{h(a)/x\}$ si ottiene dal termine $f(g(h(a)))$ la cc (b, $f(c)$). •

Vediamo un noto esempio di coppia critica generata da una sola regola il cui lato sinistro si sovrappone col lato sinistro di una variante ridenominata della stessa regola.

Esempio 22:

$$\begin{array}{rcl} f(f(x)) & \longrightarrow & r(x) \\ f(f(y)) & \longrightarrow & r(y) \end{array}$$

Mediante la sostituzione mgu $\theta = \{f(x)/y\}$, il termine $f(f(x))$ unifica con $f(y)$ dando luogo al termine unificato $f(f(f(x)))$, dal quale si ottiene la cc

$(r(f(x)), f(r(x)))$, dove $r(f(x))$ deriva dal redesso $f(f(f(x)))$ istanziando la x della regola con $f(x)$, mentre $f(r(x))$ deriva dal redesso interno $f(f(x))$ di $f(f(f(x)))$ ridenominando la y della seconda regola con x. •

Definizione 30 *$(s,\ t)$ è una* coppia critica *per un sistema di riscrittura R se esistono due regole (che non abbiano variabili in comune)*

$$
\begin{array}{ccc}
l_1 & \longrightarrow & r_1 \\
l_2 & \longrightarrow & r_2
\end{array}
$$

tali che esiste una posizione $p \in Pos'(l_1)$ ed una mgu θ tali che

$$\theta(l_1|_p) = \theta(l_2)$$

cosicché $s = \theta(l_1)[\theta(r_2)]_p$ e $t = \theta(r_1)$.

Graficamente si ha quindi:

$$
\begin{tikzcd}
& \theta(l_1) \arrow[dl] \arrow[dr] & \\
\theta(l_1)[\theta(r_2)]_p & & \theta(r_1)
\end{tikzcd}
$$

L'insieme CC delle coppie critiche di un sistema di riscrittura è formato da tutte le cc che si possono ottenere dalle sue regole e dalle varianti ridenominate delle stesse considerate due a due. Come abbiamo visto dall'Esempio 22, dopo ridenominazione, una regola si può sovrapporre anche con se stessa e generare una coppia critica.

Definizione 31 *Un sistema di riscrittura R è detto* ambiguo *se il suo insieme di coppie critiche CC è non vuoto.*

Ogni cc $(s,\ t)$, per come è stata ottenuta da R, risulta formata da due termini convertibili rispetto ad R, per cui $(s = t)$, ma non è sempre convergente.

Definizione 32 *Una cc $(s,\ t)$ di un sistema di riscrittura R è* convergente *se $s \downarrow t$.*

La rilevanza della nozione di coppia critica risiede nel fatto intuitivo che ogniqualvolta un termine t si riduce in un sol passo, mediante le regole di un sistema R, a due termini sintatticamente diversi t_1 e t_2, si ha che t_1 ha come sottotermine una istanza di un elemento di una cc di R e t_2 ha come sottotermine una istanza dell'altro elemento della stessa cc. Questo è quanto afferma il cosidetto **Lemma delle coppie critiche**, la cui dimostrazione dettaglia i vari casi sintatticamente possibili, che omettiamo di dimostrare [1].

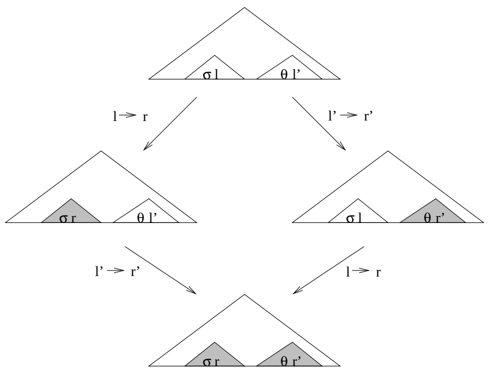
6.3 Lemma di Huet

La costruttività del seguente lemma risiede nel fatto che ogni insieme CC ha cardinalità finita.

Lemma 2 *(Lemma di Huet) Un sistema di riscrittura R è WCR se e solo se tutte le sue coppie critiche sono convergenti.*

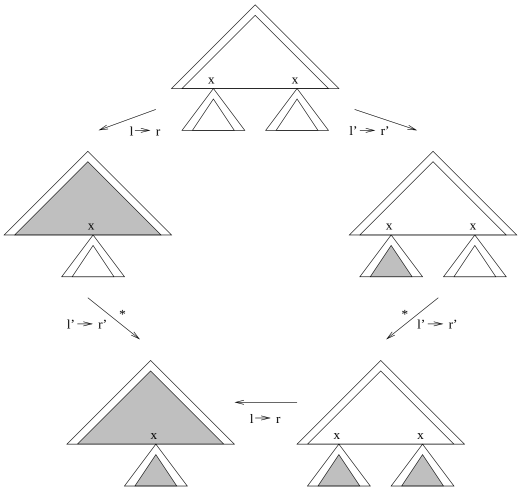
Dimostrazione Se R è WCR ovviamente tutte le sue cc sono convergenti, per cui dimostriamo soltanto il caso 'se', ossia che se tutte le sue cc sono convergenti allora R è WCR.
I seguenti tre casi possibili rispecchiano le tre possibili relazioni in cui possono trovarsi due redessi all'interno di uno stesso termine. Inoltre la rappresentazione grafica semplifica e rende intuitiva la trattazione.

1. Caso dei redessi disgiunti



Ad esempio, se il termine $t = f(g(x),\, h(b))$ si riduce in base alle regole $g(x) \longrightarrow a$ e $h(y) \longrightarrow c$, si ottengono $f(a,\, h(b))$ e $f(g(x),\, c)$ rispettivamente, che chiaramente convergono a $f(a,\, c)$.

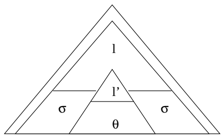
2. Caso dei redessi annidati (sovrapposizione su variabile)



Ad esempio, se il termine $t = h(f(g(x)), c)$ si riduce in base alle regole $g(y) \longrightarrow a$ e $f(x) \longrightarrow r(x, x)$, si ottengono $h(f(a), c)$ e $h(r(g(x), g(x)), c)$ rispettivamente che chiaramente convergono a $h(r(a, a), c)$.

3. Caso dei redessi sovrapposti (in cui si utilizza l'ipotesi di convergenza delle cc)

Il termine $t = \mathcal{C}[(\sigma l)[\theta l']_p]$ può essere rappresentato nel modo seguente:



Il termine t si riduce tramite la regola $l \longrightarrow r$ in $t_1 = \mathcal{C}[\sigma r]$ e tramite la regola $l' \longrightarrow r'$ in $t_2 = \mathcal{C}[(\sigma l)[\theta r']_p]$. Ponendo $\rho = \sigma \circ \theta$, si ha $t_1 = \mathcal{C}[\rho r]$ e $t_2 = \mathcal{C}[(\rho l)[\rho r']_p]$. Per l'ipotesi di convergenza delle coppie critiche, si ha $t_1 \stackrel{*}{\longrightarrow} \mathcal{C}[t'] \stackrel{*}{\longleftarrow} t_2$ per un qualche termine t'.

Ad esempio, se il termine $t = h(f(f(f(a))))$ si riduce in base alle regole $f(f(x)) \longrightarrow b$ e $f(b) \longrightarrow c$, si ottengono $h(b)$ e $h(f(b))$, che formano una cc non convergente, potendosi ottenere ancora soltanto $h(c)$ da $h(f(b))$.

Se il sistema di riscrittura possedesse ulteriori regole che rendono la cc convergente (ad esempio $b \longrightarrow c$) il sistema sarebbe WCR.

■

7 Sistemi di riscrittura canonici

Passiamo a parlare di un metodo per rendere *canonico*, detto anche *convergente* o *completo*, un sistema di riscrittura che non lo è ma che ammette un sistema finito canonico ad esso equivalente. Il significato semantico di tale metodo viene messo in evidenza dal fatto che il metodo per il completamento fornisce una procedura di decisione per il cosidetto *problema della parola (word problem)*.

7.1 Il problema della parola

Data una teoria equazionale E, il problema della parola consiste nel decidere $E \vdash s = t$, ossia se $s = t$ in base soltanto ad E. Tale problema può essere affrontato mediante riscrittura se esiste un sistema di riscrittura R equivalente ad E, ovvero con lo stesso potere deduttivo di E. Ricordiamo che un'equazione denota uguaglianza e quindi può essere applicata in entrambe le direzioni, mentre una regola di riscrittura può essere applicata in una sola direzione. Inoltre, le sole sostituzioni richieste nella riscrittura sono istanziazioni applicate alle regole.

Definizione 33 *Un sistema di riscrittura R è* equivalente *ad un insieme di assiomi equazionali E se per ogni s, $t \in Ter$ si ha*

$$s \overset{*}{\longleftrightarrow}_R t \text{ se e solo se } E \vdash s = t.$$

L'obiettivo è dunque trovare (se esiste) un sistema di riscrittura R *equivalente* ad E.

Teorema 8 *Se R equivalente ad E è canonico (o convergente o completo), allora il problema della parola rispetto ad E è decidibile.*

Dimostrazione Infatti, $s{\downarrow} \equiv t{\downarrow}$ se e solo se $s \overset{*}{\longleftrightarrow}_R t$ se e solo se $E \vdash s = t$.

■

Quindi possiamo usare $\longrightarrow$ invece di $\longleftrightarrow$ senza fare mai backtracking nelle riduzioni.

Il viceversa di questo teorema non è vero. Esistono teorie equazionali con problema della parola decidibile per le quali non esiste un sistema di riscrittura finito equivalente e completo [4].

7.2 Teorema di Knuth-Bendix

È noto come **teorema di Knuth-Bendix** il seguente risultato, che possiamo considerare corollario del lemma di Newman e del lemma di Huet:

Teorema 9 *Un sistema di riscrittura terminante è canonico se tutte le sue coppie critiche sono convergenti.*

Dimostrazione Se il sistema è SN e tutte le sue cc sono convergenti (in particolare CC $= \emptyset$), esso è WCR per il lemma di Huet. Per il lemma di Newman, SN e WCR implicano CR. ■

Dato un insieme di assiomi che specificano una particolare struttura, si tratta di determinare un sistema di riscrittura canonico, se esiste, equivalente all'insieme dato. Dato un sistema di riscrittura R non completo, il problema del suo completamento consiste dunque nel trovare un insieme R' di regole di riscrittura che sia equivalente e completo.

Vediamo come si può procedere per determinare, se esiste, un tale sistema di riscrittura canonico.

Supponiamo che sia dato un sistema di riscrittura terminante per cui affrontiamo il problema della confluenza. Da [2] possiamo formulare il problema come segue: data una relazione di riscrittura $\longrightarrow_R$, determinare una relazione di riscrittura $\longrightarrow_{R'}$ tale che $\stackrel{*}{\longleftrightarrow}_R = \stackrel{*}{\longleftrightarrow}_{R'}$ e $\longrightarrow_{R'}$ ha la proprietà CR.
L'idea è quella di rilevare tutte le possibili situazioni nelle quali viene compromessa la confluenza, ed in tali casi si propone di rimediare introducendo nella relazione R' la nuova coppia (s, t), ovvero introducendo la nuova regola $s \longrightarrow t$ oppure $t \longrightarrow s$ nella relazione $\longrightarrow_{R'}$. La coppia viene introdotta solo se la sua introduzione non pregiudica la proprietà di terminazione del sistema che si sta costruendo, mentre in ogni caso l'introduzione della nuova regola preserva la correttezza della procedura (s e t derivano entrambi da uno stesso termine) poiché $\stackrel{*}{\longleftrightarrow}_R = \stackrel{*}{\longleftrightarrow}_{R'}$. Se la procedura termina con successo nel modo che preciseremo, allora R' risulterà confluente.

Se R' è terminante, sappiamo che basta ottenere la proprietà WCR, per cui basta calcolare tutte le coppie critiche (che sono in numero finito) ed orientarle in modo compatibile con l'ordinamento di riduzione adottato per assicurarsi di conservare la terminazione.

7.3 Procedure di completamento

In base al teorema di Knuth-Bendix, sono state formulate varie procedure piú o meno sofisticate che, sotto ipotesi SN, tentano di eliminare tutte le coppie critiche di un dato sistema di riscrittura. Riportiamo le regole che bisogna in ogni caso applicare non deterministicamente, ossia scegliendo in modo arbitrario dove e quale regola applicare. Ogni strategia di applicazione delle regole (con l'eventuale aggiunta di altre regole) determina una particolare procedura di completamento.

Si parte direttamente da un insieme di assiomi equazionali E che si vuol trasformare in un sistema di riscrittura equivalente e canonico, all'inizio vuoto, ossia si parte dalla coppia $(E,\ \emptyset)$.
Si sceglie un ordinamento di riduzione $\succ$ da usare nel modo precisato in alcune delle regole in Fig. 2 al fine di conservare come abbiamo detto la proprietà di terminazione. Va tenuta presente la seguente osservazione:

Osservazione 5: Ogni procedura di completamento dipende in modo esplicito dall'ordinamento di riduzione adottato che essa deve ricevere in input insieme ad E. Si parla infatti anche dell'input $(E,\ \succ)$. ∎

Le regole basilari sono date in Fig. 2. Applicando tali regole, si costruisce man mano la successione

$$(E, \emptyset), (E_1, R_1), \ldots, (E_i, R_i), \ldots$$

dove $(E_i,\ R_i) \vdash (E_{i+1},\ R_{i+1})$ per ogni indice i che in essa occorre. Tale successione può

1. terminare con $(\emptyset,\ R_n)$. In tal caso R_n è canonico e tutte le equazioni sono state orientate.
   Se E oltre ad essere finito è anche chiuso e $\succ$ è totale sui termini chiusi, la procedura di completamento termina con successo [7, 14].

2. terminare con $(E_i,\ R_i)$ ed $E_i \neq \emptyset$, ossia fallire. Ciò si verifica ad esempio quando l'ordinamento usato non riesce ad orientare una coppia critica, il che può avvenire anche se esiste un sistema di riscrittura completo

Orienta (con analoga regola se $r \succ l$):

$$\frac{(E \cup \{l = r\}, R),\ l \succ r}{(E, R \cup \{l \longrightarrow r\})}$$

Fallimento:

$$\frac{(E \cup \{l = r\}, R),\ l \not\succ r,\ r \not\succ l}{(fallimento)}$$

Cancella:

$$\frac{(E \cup \{l = l\}, R)}{(E, R)}$$

Normalizza a sinistra (con analoga a destra):

$$\frac{(E \cup \{l = r\}, R),\ l \stackrel{+}{\longrightarrow} l\!\downarrow}{(E \cup \{l\!\downarrow = r\}, R)}$$

Uguaglia:

$$\frac{(E, R),\ (c = c') \in CC}{(E \cup \{c = c'\}, R)}$$

Figura 2: Regole basilari di completamento.

ed equivalente ad E. Per questi aspetti vedasi [4]. Cambiare ordinamento nel corso del completamento con uno 'piú potente' (che quindi relaziona anche i termini già relazionati) può rendere non corretto il completamento anche qualora termini con successo [13].

3. essere infinita e al limite fallire oppure al limite costruire un sistema di riscrittura canonico con un numero infinito numerabile di regole.

Alle regole basilari del completamento possono essere aggiunte le seguenti regole di inferenza, che consentono di semplificare le regole del sistema corrente R_i:

Componi:

$$\frac{(E, R \cup \{l \longrightarrow r\}),\ r \longrightarrow r'}{(E, R \cup \{l \longrightarrow r'\})}$$

Collassa:

$$\frac{(E, R \cup \{l \longrightarrow r\}),\ l \longrightarrow l' \ \textit{tramite}\ l_k \longrightarrow r_k \ \textit{con}\ l \longrightarrow r \rhd l_k \longrightarrow r_k}{(E \cup \{l' = r\}, R)}$$

dove $\rhd$ è un ordinamento benfondato sulle regole di riscrittura definito nel modo seguente: $l \longrightarrow r \rhd l_k \longrightarrow r_k$ se (i) un sottotermine di l è un'istanza di l_k oppure (ii) l ed l_k sono uguali ed $r \succ r_k$.
In modo simile alla regola **Normalizza** che normalizza i lati delle equazioni in E_i rispetto ad R_i, le regole **Componi** e **Collassa** permettono di ridurre le regole di R_i ottenendo cosí un sistema di regole *intraridotto*.

7.4 Proprietà delle regole di completamento

1. Le regole di completamento generano un sistema terminante.
   Infatti, ogni equazione diviene regola soltanto se l'ordinamento di riduzione $\succ$ assunto lo consente.

2. Le regole sono *corrette* nel senso che se $(E_i,\ R_i) \vdash (E_{i+1},\ R_{i+1})$, allora $\overset{*}{\longleftrightarrow}_{(E_i,R_i)} = \overset{*}{\longleftrightarrow}_{(E_{i+1},R_{i+1})}$, e quindi la teoria equazionale sottostante non risulta modificata.
   Esaminando ciascuna regola, risultano ovviamente corrette le regole **Orienta**, **Cancella** e **Uguaglia** e le rimanenti lo sono essenzialmente in base alla definizione della relazione di riscrittura.
   Non sottilizziamo sulla correttezza del completamento all'infinito (questo aspetto può essere approfondito consultando [1]) in quanto siamo qui interessati soltanto a sistemi di riscrittura finiti.

Esempio 23: L'esempio classico di Knuth-Bendix si ottiene orientando la teoria

$$\begin{aligned} 0 + x &= x \\ x + (-x) &= 0 \\ (x + y) + z &= x + (y + z) \end{aligned}$$

rispetto ad un rpo basato sulla precedenza $+ > 0$ e con l'associatività del "+" (terza equazione) orientata da sinistra a destra, ottenendo cosí il sistema incompleto

$$0 + x \longrightarrow x$$

$$
\begin{array}{rcl}
x+(-x) & \longrightarrow & 0 \\
(x+y)+z & \longrightarrow & x+(y+z)
\end{array}
$$

Infatti, ad esempio da $0+(-0)$ si ottiene la cc $(-0,0)$ non convergente; da $(x+(-x))+y$ si ottiene la cc non convergente $(y,\, x+(-x+y))$. •

Altri esempi sono riportati in [1]. Si tenga presente che gli esempi andrebbero, come di solito avviene, fatti elaborare da una procedura implementata, dato il possibile alto numero di applicazione delle regole che di solito si rende necessario. Noti sistemi implementati che contengono procedure di completamento sono alcuni sistemi di dimostrazione automatica di teoremi, quali Otter, RRL, EQP e Larch Prover [17], ed altri sistemi di dimostrazione, quali CiME, ELAN e Maude [15, 16, 18].

Esistono anche estensioni del completamento che rimuovono casi tipici di non terminazione o di fallimento. Sono note quelle che orientano in entrambi i sensi le coppie critiche non orientabili e quelle che completano modulo classi di equivalenza costruite mediantealcune equazioni di E conservate tali.

8 Riscrittura modulo equazioni

Alcune equazioni non sono orientabili, se non in modo del tutto fittizio. Si passa in tal caso ad istanziare, unificare, ... modulo tali uguaglianze di un insieme E. La riscrittura modulo (le uguaglianze assunte) rimpiazza la relazione R con R/E ovunque e una riduzione si può scrivere come segue:

$$t_1 \longrightarrow_{R/E} t_2 =_E t_2' \longrightarrow_{R/E} \dots$$

Se le classi di equivalenza sono finite, la relazione $\longrightarrow_{R/E}$ è decidibile se $\longrightarrow_R$ lo è.

8.1 E-unificazione

Ora manteniamo l'insieme E come insieme di assiomi e formuliamo il problema dell'unificazione come $s =_E t$, le cui soluzioni sono in Ter/E. L'unificazione sintattica si ha nel caso particolare di $E = \emptyset$. L'E-unificazione viene anche detta unificazione semantica.

Le nozioni tipiche dell'unificazione vengono riformulate come segue, iniziando dall'ordinamento di sussunzione sui termini e sulle sostituzioni.

Definizione 34 *Siano* s *e* t *termini in* Ter *e* σ, δ, γ, τ *sostituzioni.*

- $s \leq_E t$ *se esiste* σ *tale che* $\sigma(s) =_E t$. *La sostituzione* σ *viene detta* sostituzione istanziatrice modulo E *o* E-match *di* s *e* t.
- $\sigma \leq_E \delta$ *se esiste* γ *tale che* $\gamma(\sigma(x)) =_E \delta(x)$ *per ogni* x.
- $\sigma =_E \delta$ *se* $\sigma(x) =_E \delta(x)$ *per ogni* x.
- $\sigma \approx_E \delta$ *se* $\sigma \leq_E \delta$ *e* $\delta \leq_E \sigma$.

Osserviamo che $\sigma \leq_E \delta$ e $\delta \leq_E \sigma$ in generale non implica $\sigma =_E \delta$ come mostra il seguente esempio.

Esempio 24: Siano $E = \{f(x,x) = x\} = I$ (idempotenza), $\sigma = \{y/x\}$ e $\delta = \{f(y,z)/x\}$.
Si ha che $\sigma \approx_E \delta$ in quanto $\sigma \leq_E \delta$ con $\gamma = \{f(y,z)/y\}$ e $\delta \leq_E \sigma$ con $\gamma = \{y/z\}$, ma $\sigma(x) \neq_E \delta(x)$. •

Definizione 35 *Due termini* s *e* t *in* Ter *sono* E-unificabili *se e solo se esiste una sostituzione* σ *tale che* $\sigma(s) =_E \sigma(t)$. *La sostituzione* σ *viene detta* E-unificatrice *di* s *e* t.

Poiché non esiste in generale un'unica (a meno di ridenominazioni di variabili) E-mgu, dobbiamo considerare insiemi di unificatrici come segue.

Definizione 36 *Siano* s *e* t *termini in* Ter *e sia* U *un insieme di loro E-unificatrici.*

- U *è minimale, notazione* MU, *se per ogni* σ, $\sigma' \in U$

$$\sigma \leq_E \sigma' \Longrightarrow \sigma =_E \sigma'.$$

- U *è completo, notazione* CU, *se per ogni unificatrice* δ *di* s *e* t *esiste* $\sigma \in U$ *tale che* $\sigma \leq_E \delta$.
- U *è* CMU *se è minimale e completo.*

Tutti i semigruppi idempotenti non hanno insiemi completi di E-mgu.

Esempio 25:

- Siano $E = \{x+y = y+x\} = C$ (commutatività), $s = 0+x$ e $t = y+z$. $0 + x =_E y + z$ mediante $\theta_1 = \{0/y, x/z\}$. Da $x + 0 =_E y + z$ anche $\theta_2 = \{x/y, 0/z\}$ unifica con $\theta_1 \not\leq_E \theta_2$ e $\theta_2 \not\leq_E \theta_1$; inoltre, per ogni altra unificatrice δ (ad esempio $\delta = \{0/x, 0/y, 0/z\}$) si ha $\theta_1 \leq_E \delta$ e $\theta_2 \leq_E \delta$. Quindi $\{\theta_1,\, \theta_2\}$ è CMU.

- Siano $E = \{x + (y + z) = (x + y) + z\} =$ A (associatività), $s = x + 1$ e $t = 1 + x$.
  $CMU = \{\theta_1 = \{1/x\},\, \theta_2 = \{1 + 1/x\},\, \ldots,\, \theta_n = \{1 + 1 + \ldots + 1/x\},$ $\ldots\}$ è un insieme infinito.

- (esempio di Huet)
  Siano $E = \{f(0, x) = 0,\ g(f(y, x)) = g(x)\}$, $s = g(x)$ e $t = g(n)$, n naturale qualsiasi.
  $CMU = \emptyset$ in quanto al finito non esiste alcuna E-mgu. Infatti:
  $\sigma_1 = \{n/x\}$ unifica ma dal secondo assioma di E si ha che $n =_E f(y, n)$ per cui anche $\sigma_2 = \{f(y, n)/x\}$ unifica, con $\sigma_1 \geq_E \sigma_2$. Iterando si ha $\sigma_2 \geq_E \sigma_3 = \{f(y, f(y, n))/x\}$ e cosí via, ottenendo la successione

  $$\sigma_1 \geq_E \sigma_2 \geq_E \sigma_3 \geq_E \ldots$$

  ossia

  $$n \geq_E f(y, n) \geq_E f(y, f(y, n)) \geq_E \ldots$$

  che non ha minimo al finito.

•

L'E-unificazione è indecidibile. Un noto esempio al riguardo sono le equazioni diofantee $p_1(x) =_E p_2(x)$ dove i polinomi $p_i(x)$ $(i = 1, 2)$ hanno coefficienti in $\mathbf{Z}$ e si vogliono le soluzioni in $\mathbf{Z}$. Si tratta del problema noto come 'Decimo problema di Hilbert', risolto negativamente nel 1970 da Matijasevic.
Molti problemi sulla decidibilità della E-unificazione sono aperti. Vi sono vari algoritmi di E-unificazione per classi di assiomi di vario tipo. Un caso abbastanza generale è il seguente:
se E ammette un sistema di riscrittura canonico nel senso precedentemente chiarito, allora l'E-unificazione è semidecidibile (ossia in alcuni casi la procedura può non terminare). Occorre a questo punto la nozione di 'narrowing'.

8.2 Narrowing

Se invece di istanziare una regola di riscrittura su un termine riducibile da riscrivere, si unifica tale termine col lato sinistro della regola e poi si applica la regola, si ottiene la relazione di narrowing, notazione $\rightsquigarrow$. Illustriamo la nozione mediante un esempio.

Esempio 26: Il termine $h(f(y, a))$ risulta irriducibile rispetto alla regola $f(a, x) \longrightarrow g(b)$, ma unificando $f(y, a)$ col lato sinistro della regola restringiamo i valori di y al valore a e il termine ristretto ottenuto $(h(f(a, a)))$ può essere riscritto (in $h(g(b))$). •

In termini di linguaggi di programmazione, siamo usciti dalla programmazione funzionale per andare verso la programmmazione logica (vedi Sezione 8.5).

Definizione 37 *(narrowing) Sia dato un termine* $t \in Ter$*. Siano* $p \in Pos'(t)$ *ed* $l \longrightarrow r$ *una regola di* R *tale che* $t|_p$ *sia unificabile con* l*, ossia esiste una mgu* σ *tale che* $\sigma(l) = \sigma(t|_p)$*. Diciamo che il termine* t si riscrive via narrowing *nel termine* $t' = \sigma(t)[\sigma(r)]_p = \sigma(t[r]_p)$ *e scriviamo* $t \rightsquigarrow t'$*, dove* $\rightsquigarrow$ *è la relazione di narrowing su* Ter*.*

Si indica con $||(s, t)$ un nuovo termine, con $||$ non appartenente alla segnatura del linguaggio che si considera, dove il narrowing viene applicato indifferentemente ad s oppure a t, potendo in tal modo scrivere

$$||(s, t) \overset{*}{\rightsquigarrow} ||(s_1, t_1) \overset{*}{\rightsquigarrow} ||(s_2, t_2) \overset{*}{\rightsquigarrow} \ldots$$

Spesso le riduzioni mediante narrowing non terminano. Essendo il grafo di riduzione col narrowing ampio, si sono avute varie ottimizzazioni che non perdono la completezza.

- Narrowing *basilare*
  Il narrowing non viene applicato ad un termine che sia stato ottenuto mediante applicazione precedente di narrowing.

- Narrowing *normale*
  Il narrowing viene applicato soltanto a termini che siano in forma normale rispetto alla riscrittura.

Qui consideriamo il narrowing normale.

8.3 Procedura di E-unificazione

L'idea di base della procedura di E-unificazione di Fay-Hullot-Lankford è che in

$$||(s,t) \overset{*}{\rightsquigarrow} ||(s_1,t_1) \overset{*}{\rightsquigarrow} ||(s_2,t_2) \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow} ||(s_n,t_n)$$

se s_n e t_n sono sintatticamente unificabili, allora s e t sono E-unificabili mediante la sostituzione composta dalle n sostituzioni unificatrici del narrowing e dalla unificatrice sintattica, composte nell'ordine in cui sono state applicate.

I dati di partenza della procedura sono il sistema di riscrittura canonico R equivalente alla teoria equazionale E e $(s =_E t,\ id)$, scritto $(||(s,\ t),\ id)$. Sia σ_i l'i-esima sostituzione composta di narrowing normale, per cui al passo i-esimo si ha $(||(s_i,\ t_i),\ \sigma_i)$. I passi della procedura sono i seguenti:

$||(s_i,\ t_i)$ in forma normale rispetto ad R è tale che s_i e t_i sono unificabili con mgu θ.
Allora $\theta \circ \sigma_i$ è una E-mgu di s e t.

$||(s_i,\ t_i)$ in forma normale rispetto ad R non è unificabile con il lato sinistro di alcuna regola di R.
Allora s_i e t_i non sono E-unificabili.

$||(s_i,\ t_i)$ in forma normale rispetto ad R è riscrivibile via narrowing con una regola di R mediante σ.
Allora $(||(s_i,\ t_i),\ \sigma_i)$ viene aggiornato con $(||(s_{i+1},\ t_{i+1}),\ \sigma \circ \sigma_i)$.

Tale procedura enumera le E-mgu di due termini e termina se tali termini sono E-unificabili. Il risultato (di cui omettiamo la dimostrazione) che garantisce la correttezza e la completezza di tale procedura è:

Teorema 10 *Sia data una teoria equazionale E che ammette un sistema di riscrittura canonico R e siano dati i termini s e t. Consideriamo il termine ausiliario $||(s,\ t)$. Se in*

$$||(s,t) \overset{*}{\rightsquigarrow} ||(s_1,t_1) \overset{*}{\rightsquigarrow} ||(s_2,t_2) \overset{*}{\rightsquigarrow} \ldots \overset{*}{\rightsquigarrow} ||(s_n,t_n)$$

$||(s_n,\ t_n)$ è tale che s_n e t_n sono sintatticamente unificabili mediante l'mgu θ, allora s e t sono E-unificabili mediante la sostituzione composta dalle n sostituzioni unificatrici del narrowing e dalla unificatrice sintattica θ (nell'ordine in cui sono state applicate).
L'insieme delle sostituzioni trovate in tal modo risulta un insieme completo ma non minimale di E-unificatrici di s e t.

Esempio 27: Sia data la teoria equazionale E:

$$\begin{aligned} +(0,x) &= x \\ +(s(x),y) &= s(+(x,y)) \end{aligned}$$

Applicando la procedura di completamento, troviamo che il sistema canonico equivalente ad E è dato semplicemente dalle due regole ottenute orientando da sinistra a destra le equazioni di E rispetto ad un ordinamento rpo basato sulla precedenza $+ > s$:

$$\begin{aligned} +(0,x) &\longrightarrow x \\ +(s(x),y) &\longrightarrow s(+(x,y)) \end{aligned}$$

Supponiamo di voler risolvere l'equazione $+(x,x) = s(s(0))$ modulo E utilizzando la procedura di E-unificazione basata su narrowing normale. Il termine iniziale (detto anche *goal*) è scritto $||(+(x,x),\ s(s(0)))$. I termini dell'equazione sono in forma normale rispetto ad R. A partire dal goal iniziale sono possibili le seguenti riscritture via narrowing, entrambe alla posizione $p = 1$:

$$\begin{aligned} ||(+(x,x),s(s(0))) &\leadsto ||(0,s(s(0))) \\ ||(+(x,x),s(s(0))) &\leadsto ||(s(+(x_1,s(x_1))),s(s(0))) \end{aligned}$$

tramite, rispettivamente, la prima regola (ridenominata) $+(0,x_1) \longrightarrow x_1$ con sostituzione unificatrice $\sigma_1 = \{0/x, 0/x_1\}$, e la seconda regola (ridenominata) $+(s(x_1),y_1) \longrightarrow s(+(x_1,y_1))$ con sostituzione unificatrice $\sigma_1 = \{s(x_1)/x, s(x_1)/y_1\}$.
Il termine $||(0,s(s(0)))$ (in forma normale rispetto ad R) non è unificabile con alcuna delle regole di R, né i due termini 0 e $s(s(0))$ sono unificabili sintatticamente, quindi si ha fallimento. Il termine $||(s(+(x_1,s(x_1))),s(s(0)))$ (in forma normale rispetto ad R) è unificabile alla posizione $p = 1.1$ con il lato sinistro di entrambe le regole di R dando luogo alle seguenti riscritture via narrowing:

$$\begin{aligned} ||(s(+(x_1,s(x_1))),s(s(0))) &\leadsto ||(s(s(0)),s(s(0))) \\ ||(s(+(x_1,s(x_1))),s(s(0))) &\leadsto ||(s(s(+(x_2,s(s(x_2))))),s(s(0))) \end{aligned}$$

tramite, rispettivamente, la regola $+(0,x_2) \longrightarrow x_2$ con sostituzione unificatrice $\sigma_2 = \{0/x_1, s(0)/x_2\}$, e la regola $+(s(x_2),y_2) \longrightarrow s(+(x_2,y_2))$ con sostituzione unificatrice $\sigma_2 = \{s(x_2)/x_1, s(s(x_2))/y_2\}$.

I due termini del goal corrente $||(s(s(0)), s(s(0)))$ unificano sintatticamente con mgu $\theta = id$. Componendo le sostituzioni unificatrici σ_1, σ_2 e θ, otteniamo la soluzione $x = s(x_1) = s(0)$. Infatti, se sostituiamo x con $s(0)$ nell'equazione iniziale $+(x, x) = s(s(0))$, abbiamo $+(s(0), s(0)) = s(s(0))$ che risulta vera modulo E (la forma normale rispetto ad R di $+(s(0), s(0))$ è uguale a $s(s(0))$).
A partire dall'altro goal corrente $||(s(s(+(x_2, s(s(x_2))))), s(s(0)))$ è invece possibile applicare ulteriori riscritture via narrowing alla posizione $p = 1.1.1$ e successivamente in posizioni sempre piú annidate nei termini risultanti senza ottenere altre soluzioni per l'equazione iniziale.
L'albero delle derivazioni di narrowing è il seguente:

$$
\begin{tikzcd}[column sep=small]
& ||(+(x,x), s(s(0))) \arrow[dl] \arrow[dr] & & & \\
Fallimento & & ||(s(+(x_1, s(x_1))), s(s(0))) \arrow[dl] \arrow[dr] & & \\
& Successo & & ||(s(s(+(x_2, s(s(x_2))))), s(s(0))) \arrow[dl] \arrow[dr] & \\
& & Fallimento & & \ldots
\end{tikzcd}
$$

•

L'insieme delle E-unificatrici risulta in generale completo ma non minimale.

Nella Sezione 8.2 abbiamo accennato ad una particolare forma di narrowing, detta *basilare*, in cui le riscritture via narrowing non vengono applicate a quei termini che sono stati ottenuti mediante un'applicazione precedente di narrowing. Ciò porta ad una riduzione dello spazio di ricerca della procedura di E-unificazione senza perdere correttezza e completezza [10]. La procedura viene modificata restringendo le posizioni del termine in cui si possono applicare i passi di narrowing. Le posizioni basilari sono un sottoinsieme delle posizioni non di variabili, considerate nella Definizione 37, e sono definite nel modo seguente. Dato il termine iniziale $||(s, t)$, l'insieme delle posizioni pos_0 coincide con $Pos'(||(s, t))$, mentre alle iterazioni successive, dato il termine $||(s_i, t_i)$ ottenuto mediante l'applicazione di un passo di narrowing alla posizione p usando la regola $l \longrightarrow r$, si ha:

$$pos_{i+1} = (pos_i \setminus \{q \mid p \leq q\}) \cup \{p.q \mid q \in Pos'(r)\}$$

Quindi, le posizioni del termine piú *basse* della posizione p (inclusa p) vengono rimpiazzate dalle posizioni non di variabili del lato destro (non istanziato) della regola usata al passo precedente di narrowing.

8.4 Quadro relativo alle teorie equazionali

In letteratura è stato considerato il problema dell'unificazione modulo varie teorie equazionali particolari. Le teorie equazionali piú note sono le seguenti:

- $E = \{f(x, y) = f(y, x)\} = C$ (commutatività)
- $E = \{f(x, f(y, z)) = f(f(x, y), z)\} =$ A (associatività)
- $E = A \cup C = AC$
- $E = \{f(x, x) = x\} = I$ (idempotenza)
- $E = A \cup I = AI$
- $E = \{g(x, f(y, z)) = f(g(x, y), g(x, z))\} = D_s$ (distributività a sinistra)
- $E = \{g(f(y, z), x) = f(g(y, x), g(z, x))\} = D_d$ (distributività a destra)
- $E = D_s \cup D_d = D$ (distributività)
- $E = A \cup D = AD$

Dati due termini s e t ed una teoria equazionale E, indichiamo con $\mathcal{U}_E(s, t)$ un insieme completo e minimale di E-unificatrici di s e t. Con $|S|$ denotiamo la cardinalità di un insieme S. L'unificazione modulo E è:

- *unitaria* se $|\mathcal{U}_E(s, t)| \leq 1$;
- *finitaria* se $|\mathcal{U}_E(s, t)| < \omega$;
- *infinitaria* se $|\mathcal{U}_E(s, t)| = \omega$;
- *nullaria* se $\mathcal{U}_E(s, t)$ non esiste.

Sappiamo che l'unificazione sintattica ($E = \emptyset$) è unitaria (Sezione 2.5.2). Inoltre, abbiamo già visto nell'Esempio 25 che la teoria C della commutatività è finitaria e la teoria A dell'associatività è infinitaria. I risultati relativi all'unificazione modulo le teorie equazionali date sopra possono essere riassunti come segue:

- $E = \emptyset$ unitaria;
- C finitaria;
- A infinitaria;
- AC finitaria;
- I finitaria;
- AI nullaria;
- D_s unitaria;
- D_d unitaria;
- D infinitaria;
- AD infinitaria.

Per maggiori dettagli sull'E-unificazione in queste teorie equazionali e sugli algoritmi relativi rimandiamo a [1].

8.5 Confronto con la SLD-risoluzione

Vediamo, tramite un esempio, come le equazioni possono essere risolte modulo una teoria equazionale E utilizzando la SLD-risoluzione del Prolog [6, 11].

Esempio 28: Facendo riferimento all'Esempio 27, la teoria equazionale E in forma di clausole Horn è la seguente:

$$
\begin{aligned}
& & plus(0, x_1, x_1). & \qquad (4)\\
plus(s(x_2), y_2, s(z_2)) & :- & plus(x_2, y_2, z_2). & \qquad (5)
\end{aligned}
$$

L'equazione $+(x, x) = s(s(0))$ da risolvere modulo E si traduce nel goal:

$$:- \ plus(x, x, s(s(0))).$$

Tale goal unifica con la testa della clausola 5 con sostituzione unificatrice $\{s(x_2)/x, s(x_2)/y_2, s(0)/z_2\}$. Il nuovo goal è

$$:-\ plus(x_2, s(x_2), s(0)).$$

che unifica con la testa della clausola 4 con sostituzione $\{0/x_2, s(0)/x_1\}$. Si ottiene cosí la clausola vuota

$$\Box.$$

e componendo le sostituzioni si ha la sostituzione di risposta $x = s(x_2) = s(0)$.
L'albero SLD è il seguente:

$$\begin{array}{lcc}
:-\ plus(x, x, s(s(0))). & & \\
\quad\quad\searrow & & \\
& :-\ plus(x_2, s(x_2), s(0)). & \\
& \swarrow \quad \searrow & \\
& Successo \quad Fallimento &
\end{array}$$

•

Questo esempio mostra che l'E-unificazione basata su narrowing e la SLD-risoluzione sono approcci alternativi alla risoluzione di equazioni modulo una teoria equazionale. Narrowing e risoluzione sono regole di inferenza basate sullo stesso tipo di meccanismo, in quanto cercano entrambe di unificare un pezzo del goal (che nel narrowing è un sottotemine di un lato dell'equazione, mentre nella risoluzione è un letterale) con un pezzo di una regola (che nel narrowing è il lato sinistro di una regola di riscrittura, mentre nella risoluzione è la testa di una clausola). Sia il narrowing che la risoluzione applicano la sostituzione unificatrice all'intero goal e all'intera regola/clausola, e costruiscono un nuovogoal prendendo le istanze dei pezzi rimanenti del goal e della regola/clausola.

La differenza tra narrowing e risoluzione consiste nel fatto che il narrowing, nel cercare di unificare, considera tutti i possibili sottotermini ai quali possa essere applicato un passo di narrowing, mentre la risoluzione considera un intero letterale con tutti i suoi argomenti. Ne consegue che nel narrowing sono esplorate tutte le possibili scelte di sottotermini, mentre nella risoluzione sono esplorati solo i cammini che partono dall'applicazione di un passo di risoluzione al letterale selezionato. Ciò porta ad avere nel narrowing uno spazio di ricerca molto piú grande di quello che si ha nella risoluzione e quindi tempi assai piú lunghi. Negli Esempi 27 e 28 possiamo già rilevare una differenza tra gli spazi di ricerca. Al primo passo il narrowing tenta (essendo possibile) l'applicazione di entrambe le regole di R. L'applicazione della prima regola porta al fallimento mentre, una volta applicata la seconda regola, si ottiene un goal a cui è ancora possibile applicare entrambe le regole: la prima regola porta al successo, mentre la seconda porta ad un numero infinito di fallimenti e alla non terminazione della procedura di E-unificazione basata su narrowing. Tale procedura va alla ricerca di un insieme completo di E-unificatrici, senza riconoscere che tale insieme è costituito dalla sola sostituzione $\{s(0)/x\}$. La risoluzione invece al primo passo può applicare solo la clausola 5 e al goal ottenuto può applicare sia la 4 che la 5. La SLD-risoluzione del Prolog applica la 4 ed arriva alla terminazione con successo. L'eventuale applicazione della 5 porta ad un goal a cui non è possibile applicare nessuna delle due clausole, per cui si ha fallimento e terminazione.

9 Sistemi di riscrittura ortogonali

Sappiamo che un sistema di riscrittura è detto *ortogonale* se è lineare a sinistra e non ambiguo (Definizione 27). Vogliamo qui considerare alcune proprietà rilevanti di tali sistemi.

Teorema 11 *Ogni sistema di riscrittura ortogonale è Church-Rosser.*

La dimostrazione di questo teorema consiste nell'applicazione ripetuta del lemma delle mosse parallele (PML, da Parallel Moves Lemma).
Detto lemma usa la nozione di *residuo* di un redesso, che possiamo dire essere ciò che resta di un redesso, sia r, di un termine $t[r, r', \ldots]$ nel termine che si ottiene riducendo un redesso diverso da r, sia r' di t. Se si riducono n redessi diversi da r in t si ha la nozione di *residuo rispetto ad una riduzione finita* oppure *rispetto ad un insieme di redessi.*

Lemma 3 *In ogni sistema di riscrittura ortogonale si ha che se un termine $t[r, r', \ldots]$ con k redessi si riduce a t' riducendo r e a t'' riducendo tutti i rimanenti suoi redessi, t' e t'' convergono ad un termine t''' riducendo tutti e soli i residui in t' e t''.*

La dimostrazione del PML è per casi. I casi che bisogna esaminare sono quelli per i residui di un redesso:

1. r e r' sono disgiunti in t.
   In tal caso r si ritrova inalterato in t'.

2. r' è sottotermine proprio di r.
   In tal caso r si ritrova in t' con uno dei suoi sottotermini cambiato.

3. r è sottotermine proprio di r'.
   In tal caso $k \geq 0$ copie di r si ritrovano in t'.

Osserviamo il fatto che l'ortogonalità esclude che possano esservi redessi sovrapposti.

10 Strategie di riduzione

Definizione 38 *Una strategia di riduzione per un sistema di riscrittura R è ogni $F : Ter_R \longrightarrow Ter_R$ tale che*

$F(t) = t$ se t è in forma normale;

$t \longrightarrow F(t)$ altrimenti, ossia fa ottenere un ridotto di t.

10.1 Strategie sequenziali e parallele

Ovviamente si vuole che una strategia sia effettiva, ossia ricorsiva nel senso che diviene una funzione ricorsiva dopo aver effettuato una codifica dei termini nei numeri naturali. Nel seguito sottindendiamo che le strategie definite sono ricorsive.

Una tale strategia viene anche detta *sequenziale* (o a un passo) contrapposta ad una *parallela* (o a piú passi), la cui definizione differisce soltanto in quanto invece di $t \longrightarrow F(t)$ si ha $t \xrightarrow{+} F(t)$.

Definizione 39 *Dato un sistema di riscrittura R, una strategia di riduzione F è detta*

normalizzante *per R se, per ogni termine t che ha forma normale rispetto ad R, l'insieme $\{F^n(t) \mid n \geq 0\}$ contiene una forma normale (quindi la strategia è SN e l'insieme delle sue forme normali coincide con quello di $\longrightarrow$);*

cofinale *per* R *se, per ogni termine* t*, l'insieme* $\{F^n(t) \mid n \geq 0\}$ *è cofinale in* $G(t)$*, piú precisamente nell'insieme dei nodi del grafo di riduzione di* t*.*

L'importanza delle strategie cofinali consiste, oltre che nella nota equivalenza che sussiste tra CP e CR, anche nel fatto che, se ridurre significa aumentare informazione (per cui la forma normale ha il massimo di informazione), un insieme cofinale nel grafo di riduzione risulta certamente significativo da tale punto di vista informativo.

Sono state definite varie strategie di riduzione tenendo conto della posizione del redesso (o dei redessi nel caso di una strategia parallela), che si riduce (si riducono) ad ogni passo. Ne elenchiamo alcune molto usate, sequenziali e parallele.

1. F_{lo} che riduce sempre il redesso piú esterno e piú a sinistra (leftmost outermost) in un termine riducibile.

2. F_{li} che riduce sempre il redesso piú a sinistra e piú interno (leftmost innermost) in un termine riducibile.

3. F_{po} che riduce sempre tutti i redessi disgiunti piú esterni come sottotermini di un termine riducibile (parallel outermost).

4. F_{KG} che riduce sempre tutti i redessi di un termine riducibile (strategia di Knuth-Gross) prima di passare a ridurre nuovi eventuali redessi che si sono formati riducendo.

Teorema 12 *Sia* R *un sistema di riscrittura ortogonale.*

1. F_{po} *è normalizzante per* R*.*

2. F_{KG} *è cofinale per* R*.*

Ne omettiamo la dimostrazione.

Esempio 29: Rispetto al sistema di riscrittura R:

$$\begin{aligned} f(x,y) &\longrightarrow x \\ g(x) &\longrightarrow b \\ a &\longrightarrow b \end{aligned}$$

il termine $h(f(g(c), a), g(b))$ si riduce mediante F_{lo} come segue:

$$h(f(g(c), a), g(b)) \longrightarrow g(c) \longrightarrow b$$

mediante F_{li} come segue:

$$h(f(g(c), a), g(b)) \longrightarrow h(f(b, a), g(b)) \longrightarrow h(f(b, b), g(b)) \longrightarrow h(b, b)$$

mediante F_{po} come segue:

$$h(f(g(c), a), g(b)) \longrightarrow h(g(c), b) \longrightarrow h(b, b).$$

•

Definizione 40 *Un sistema di riscrittura è detto* normale a sinistra *se in ogni sua regola $l \longrightarrow r$ i simboli di funzione in l (quindi anche le costanti) precedono, nella notazione lineare del termine, le variabili.*

Esempio 30: Ogni schema di programma ricorsivo è normale a sinistra.
$f(x, a) \longrightarrow b$ non lo è. •

Teorema 13 *Sia R un sistema di riscrittura ortogonale e normale a sinistra. F_{lo} è normalizzante per R.*

Ne omettiamo la dimostrazione.

10.2 Sistemi di riscrittura sequenziali

Definizione 41 *Un sistema di riscrittura è detto* sequenziale *se ammette strategie sequenziali normalizzanti.*

Tale proprietà è indecidibile.

Definizione 42 *Un sistema di riscrittura è detto* fortemente sequenziale *se ammette strategie sequenziali normalizzanti ricorsive a* 1 *passo.*

La forte sequenzialità è decidibile.

Teorema 14 *Ogni sistema di riscrittura ortogonale e normale a sinistra è fortemente sequenziale.*

Ne omettiamo la dimostrazione.

Bibliografia

[1] F. Baader e T. Nipkow, '*Term Rewriting and All That*', Cambridge University Press, Cambridge, 1998.

[2] B. Buchberger, 'History and Basic Features of the Critical Pair / Completion Procedure', *Journal of Symbolic Computation* **3**, 1987, pp. 3–38.

[3] N. Dershowitz e J.-P. Jouannaud, 'Rewrite Systems', in J. van Leeuwen editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, North-Holland, 1990, pp. 243-320.

[4] N. Dershowitz, L. Marcus e A. Tarlecki, 'Existence, uniqueness and construction of rewrite systems', *SIAM Journal of Computing* **17**, 1988, pp. 629–639.

[5] H. Ehrig e B. Mahr, '*Fundamentals of Algebraic Specifications 1: Equations and Initial Semantics*', EATCS Monograph Series, Vol. 6, Springer-Verlag, 1985.

[6] J. H. Gallier, '*Logic for Computer Science: Foundations of Automatic Theorem Proving*', Computer Science and Technology Series, Harper and Row, 1986.

[7] J. Gallier, P. Narendram, D. Plaisted, S. Raatz e W. Snyder, 'An algorithm for finding canonical sets of ground rewriting rules in polynomial time', *Journal of ACM* **40**, 1993, pp. 1–16.

[8] G. Huet, 'Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems', *Journal of ACM* **27**, No. 4, October 1980, pp. 797–821.

[9] G. Huet, 'A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm', *Journal of Computer and System Sciences* **23**, 1981, pp. 11–21.

[10] J.-M. Hullot, 'Canonical Forms and Unification', in Proceedings of $5th$ CADE, Lecture Notes in Computer Science 87, Springer-Verlag, 1980, pp. 318–334.

[11] J. W. Lloyd, '*Foundations of Logic Programming*', seconda edizione, Springer, 1987.

[12] M. Nesi e M. Venturini Zilli, 'Sistemi di riduzione astratti', Research Report SI-98/06, Facoltà di Scienze MM. FF. NN., Università degli Studi di Roma "La Sapienza", March 1998.

[13] A. Sattler-Klein, 'About changing the ordering during Knuth-Bendix completion', *STACS 94, Symposium on Theoretical Aspects of Computer Science* **15**, 1994, pp. 175–186.

[14] W. Snyder, 'A fast algorithm for generating reduced ground rewriting systems from a set of ground equations', *Journal of Symbolic Computation* **15**, 1993, pp. 415–450.

[15] CiME, in `http://www.lri.fr/~demons/cime.html`.

[16] ELAN, in `http://www.loria.fr/ELAN`.

[17] Larch Prover, in `http://www.sds.lcs.mit.edu/spd/larch/LP`.

[18] The Maude System, in `http://maude.csl.sri.com/`.