Sistemas de reescritura desde el punto de vista de la programación funcional.

Miguel Ángel Porras Naranjo

Sistemas de reescritura desde el punto de vista de la programación funcional.

Miguel Ángel Porras Naranjo

Memoria presentada como parte de los requisitos para la obtención del título de Grado en Matemáticas por la Universidad de Sevilla.

Tutorizada por

Prof. José A. Alonso
Prof. María José Hidalgo

Índice general

Confluencia

En este capítulo estudiaremos el problema de determinar si un sistema de reescritura es confluente. En la primera sección vamos a demostrar que este problema es indecidible, sin embargo en las secciones posteriores, estudiaremos que si el sistema es terminante entonces el problema es decidible. Por último veremos que ocurre para el caso de los sistemas que no terminan.

Estudio sobre el problema de decisión

En esta sección veremos la indecibilidad para comprobar si un sistema es confluente mediante el siguiente resultado,

Teorema 1.1. *El problema de decidir si un sistema de reescritura finito R es confluente, es indecidible.*

Demostración. El objetivo de esta demostración es reducir el problema de las palabras básicas para E (que sabemos que es indecidible), a un sistema de reescritura de términos.

Sea un conjunto de identidades E tal que $\text{Var}(l) = \text{Var}(r)$ para todo $l \approx r \in E$. Sea $R := E \cup E^{-1}$, entonces al tener $\to_R = \leftrightarrow_E$ es confluente. Además R es un sistema de reescritura (por $\text{Var}(l) = \text{Var}(r)$).

Dados dos términos básicos s y t, y una constante a, vamos a probar que $R_{st} := R \cup \{a \to s, a \to t\}$ es confluente syss $s \approx_E t$.

- ($\Rightarrow$) Si R_{st} es confluente, entonces ni s, ni t poseen una constante a, luego $s \downarrow_{R_{st}} t$. Esto significa que las reglas $a \to s, a \to t$ no pueden ser usadas. Por tanto $\to_R = \leftrightarrow_E$ y $s \approx_E t$.

- ($\Rightarrow$) Vamos a probar que para términos u, v, si $u \to_{R_{st}} v$ entonces $v^t \xrightarrow{*}_R u^t$, donde u^t denota el resultado de sustituir las constantes a en u por t. Supongamos que $u \to_{R_s t} v$. Distinguimos que reglas se usan.
  - Si usamos $u \to_R v$, reemplazamos a por t para conseguir $u^t \to_R v^t$ y por tanto $v^t \to_R u^t$ al ser R simétrico.
  - Si usamos $a \to s$, entonces $u|_p = a$ y $v = u[s]_p$ para alguna posición p. Como $s \approx_E t$ entonces $s \xrightarrow{*}_R t$. Obtenemos $v \xrightarrow{*}_R u[t]_p$ que conlleva a $v^t \xrightarrow{*}_R (u[t]_p)^t$. Pero $(u[t]_p)^t = u^t[t^t]_p = u^t[t]_p = u^t$.
  - Si usamos $a \to t$ en la posición p, entonces $v^t = (u[t]_p)^t = u^t \xrightarrow{*}_R u^t$.

  Por tanto, si $u \xrightarrow{*}_{R_{st}} u_i$, para $i = 1, 2$, entonces $u_i \xrightarrow{*}_{R_{st}} u_i^t \xrightarrow{*}_R u^t$ y obtenemos $u_1 \downarrow_{R_{st}}$

Pares críticos

En esta sección estudiaremos la decibilidad para sistemas de reescritura finitos que sean localmente confluentes.

Definición 1.1. *Sea $l_i \to r_i$, $i = 1, 2$ dos reglas cuyas variables han sido renombradas tal que $Var(l_1, r_1) \cap Var(l_2, r_2) = \emptyset$. Sea $p \in Pos(l_1)$ tal que $l_1|_p$ no es una variable, y θ un umg de $l_1|_p =^? l_2$. Esto determinará el par crítico $\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$. Si dos reglas dan lugar a un par crítico, diremos que se solapan.*

Funciones auxiliares para Haskell

Usaremos las siguientes funciones de la libreria Data.List

- `(xs ++ ys)` es la concatenación de `xs` e `ys`. Por ejemplo,

```
> [2,5] ++ [3,7,6]
[2,5,3,7,6]
```

- `(any p xs)` se verifica si algún elemento de `xs` cumple la propiedad `p`. Por ejemplo,

```
> any even [3,2,5]
True
> any even [3,1,5]
False
```

- `(all p xs)` se verifica si todos los elementos de `xs` cumplen la propiedad `p`. Por ejemplo,

```
> all even [2,6,8]
True
> all even [2,5,8]
False
```

- `(null xs)` se verifica si `xs` es la lista vacía. Por ejemplo,

```
> null []
True
> null [3]
False
```

- `(zip xs ys)` es la lista de pares formado por los correspondientes elementos de `xs` e `ys`.

```
> zip [3,5,2] [4,7]
[(3,4),(5,7)]
> zip [3,5] [4,7,2]
[(3,4),(5,7)]
```