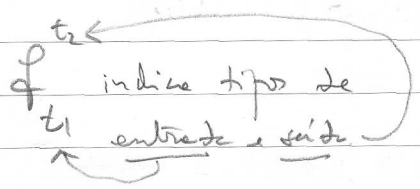
Algoritmo de Hindley-Milner - calcula o tipo mais geral de uma função por unificação

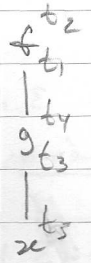
Princípio: (a) todos os tipos são distintos à partida
(b) unificam-se os tipos de acordo com as regras seguintes:

NB: notação



b.1) Aplicação:

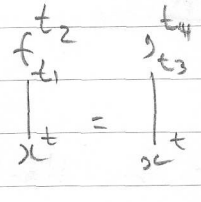
$f_{t_1}\ t_2\ t_4\ t_5\ (g\ x)_{t_1\ t_3}$



$t_1 = t_4$
$t_5 = t_3$

b.2) Igualdade (extensional)

$f^{t}_{t_1}\ x\ =\ g^{t}_{t_3}\ x$



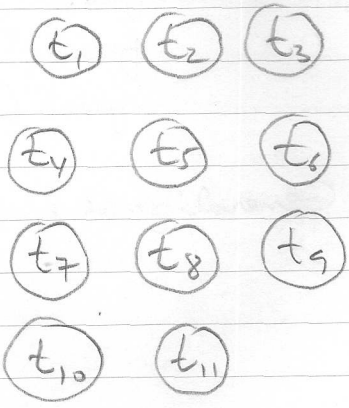
$t_1 = t$
$t_3 = t$
$t_2 = t_4$

Caso de estudo:

$f :: t_1 \to t_2$
$[\,] :: t_3$
$z :: t_4$
$(:) :: t_5 \to t_6 \to t_7$
$\oplus :: t_8 \to t_9 \to t_{10}$

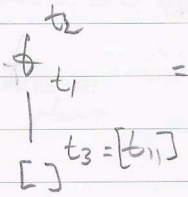
$f :: t_1 \to t_2$
$f\ [\,] = z$
$f\ (h:t) = h \oplus f\ t$

Vamos representar compatibilidade de forma gráfica



Configuração inicial:

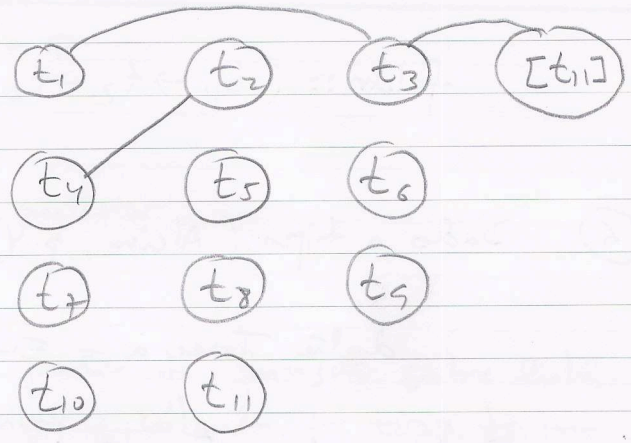
Cláusula $f\ [\,] = z$



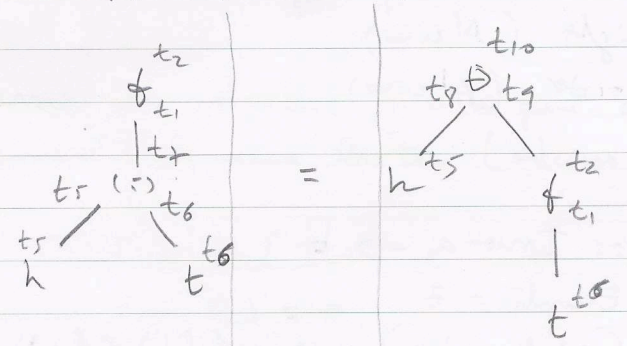
$=$ z^{t_4}

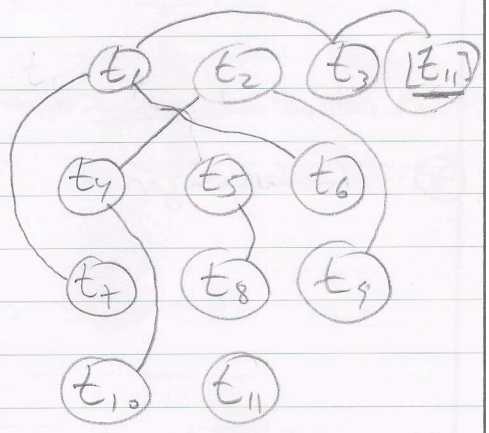
$t_2 = t_4$

$t_1 = t_3 = [t_{11}]$



Cláusula $f(h:t) = h \oplus f\,t$





$t_1 = t_7$	$t_2 = t_{10}$	$t_5 = t_8$ $t_4 = t_2$ $t_1 = t_6$

$f :: [t_{11}] \to t_2$

$[\,] :: [t_{11}]$

$z :: t_2$

$(:) :: t_5 \to [t_{11}] \to [t_{11}]$

$\oplus :: t_5 \to t_2 \to t_2$

t_{11}

Finalmente:

- Listas em Haskell são homogéneas
- Logo, $t_5 = t_{11}$

Unificação: $t_1 \cup t_3 = t_1$ (ou t_3, há que escolher)

$t_1 \cup [t_{11}] = [t_{11}]$