Peter Neubert, Leona Schlöder July 7, 2014

Hauptseminar mathematische Logik

Herbrandmodell und Unifikation

Aufgabe 1.

Aufgabe 2. Bestimme Herbranduniversum von einer gegebenen Formel:

$$\Phi_1 = \exists x \quad \phi(x)$$
$$\Phi_2 = \forall x(x = 0 \vee \exists x \quad x = y + 1)$$
$$\Phi_3 = \exists x \forall y(P(x) \rightarrow P(y))$$

Lösung: Das Herbranduniversum einer Fomel ϕ ist im Harrison definiert als die Menge bestehend aus allen Kombinationsmöglichkeiten, die aus Konstanten und Funktioen gebildet werden können. Somit ergibt sich:

- $H_{\Phi_1} = \{c, \Phi(c), \Phi(\Phi(c)), ...\}$
- $H_{\Phi_2} = \{0, 0 + 0, 0 + ... + 0, 1, 1 + 1, 1 + ... + 1, 0 + 1, 0 + 0 + 1usw\}$
- $H_{\Phi_3} = \{c\}$

Aufgabe 3. Die Formel Φ_3 hat nach Skolemisierung und Negation die folgende Form:

$$\forall x(P(x) \wedge \neg P(f_y(x))$$

Zeige an dieser Formel wie der Gilmore-Algorithmus, der den Herbrandgedanken nutzt. Füge schrittweise erst eine Konstante c, dann $f_y(c)$ usw. ein und prüfe die so entstehenden Instanzen.

Lösung: Der Algorithmus setzt die Grundinstanzen aus dem Universum in die Formel ein, bis ein Widerspruch entsteht.

1. $P(c) \wedge \neg P(f_y(c)$
2. $P(f_y(c) \wedge \neg P(fy(f_y(c))$

Der Widerspruch entsteht hier, wenn man beide Zeilen zusammen betrachtet.

Aufgabe 4. Unifiziere folgende Mengen mit Hilfe des Algorithmus oder entscheide, dass sie nicht unifizierbar sind!

1. $\{(f(x, y, z), f(g(y, y), g(z, z), a))\}$
2. $\{(f(G(a, x), y, h(a)), f(y, z, x))\}$
3. $\{(F(h(x), h(y), x, y), F(z, g(z), a, z))\}$
4. $\{(f(g(x), z, z), f(y, y, x))\}$
5. $\{(f(x, h(y), y), f(g(z), x, z))\}$

Lösung:

1.

	eqs	env
	$\{(f(x,y,z), f(g(y,y), g(z,z), a)\}$	$\{\}$
$\rightarrow$	$\{(x, g(y,y)), (y, g(z,z)), (z,a)\}$	$\{\}$
$\rightarrow$	$\{(y, g(z,z)), (z,a)\}$	$\{x \mapsto g(y,y)\}$
$\rightarrow$	$\{(z,a)\}$	$\{x \mapsto g(y,y), y \mapsto g(z,z)\}$
$\rightarrow$	$\{\}$	$\{x \mapsto g(y,y), y \mapsto g(z,z), z \mapsto a\}$
$\xrightarrow{\text{solve}}$	$\{\}$	$\{x \mapsto g(g(a,a), g(a,a)), y \mapsto g(a,a), z \mapsto a\}$

2.

	eqs	env
	$\{(f(G(a,x), y, h(a)), f(y,z,x))\}$	$\{\}$
$\rightarrow$	$\{(G(a,x), y), (y,z), (h(a), x)\}$	$\{\}$
$\rightarrow$	$\{(y, G(a,x)), (y,z), (h(a), x)\}$	$\{\}$
$\rightarrow$	$\{(y,z), (h(a), x)\}$	$\{y \mapsto G(a,x)\}$
$\rightarrow$	$\{(G(a,x), z), (h(a), x)\}$	$\{y \mapsto G(a,x)\}$
$\rightarrow$	$\{(z, G(a,x)), (h(a), x)\}$	$\{y \mapsto G(a,x)\}$
$\rightarrow$	$\{(h(a), x)\}$	$\{y \mapsto G(a,x), z \mapsto G(a,x)\}$
$\rightarrow$	$\{(x, h(a))\}$	$\{y \mapsto G(a,x), z \mapsto G(a,x)\}$
$\rightarrow$	$\{\}$	$\{y \mapsto G(a,x), z \mapsto G(a,x), x \mapsto h(a)\}$
$\xrightarrow{\text{solve}}$	$\{\}$	$\{y \mapsto G(a, h(a)), z \mapsto G(a, h(a)), x \mapsto h(a)\}$

3.

	eqs	env
	$\{(F(h(x), h(y), x, y), F(z, g(z), a, z))\}$	$\{\}$
$\rightarrow$	$\{(h(x), z), (h(y), g(z)), (x,a), (y,z)\}$	$\{\}$
$\rightarrow$	$\{(z, h(x)), (h(y), g(z)), (x,a), (y,z)\}$	$\{\}$
$\rightarrow$	$\{(h(y), g(z)), (x,a), (y,z)\}$	$\{z \mapsto h(x)\}$
$\rightarrow$	fail da $h \neq g$	

4.

eqs	env
$\{(f(g(x),z,z),f(y,y,x)\}$	$\{\}$
$\rightarrow\{(g(x),y),(z,y),(z,x)\}$	$\{\}$
$\rightarrow\{(y,g(x)),(z,y),(z,x)\}$	$\{\}$
$\rightarrow\{(z,y),(z,x)\}$	$\{y \mapsto g(x)\}$
$\rightarrow\{(z,x)\}$	$\{y \mapsto g(x), z \mapsto y\}$
$\rightarrow\{(y,x)\}$	$\{y \mapsto g(x), z \mapsto y\}$
$\rightarrow\{(g(x),x)\}$	$\{y \mapsto g(x), z \mapsto y\}$
$\rightarrow\{(x,g(x))\}$	$\{y \mapsto g(x), z \mapsto y\}$
$\rightarrow$fail "'cyclic"'	

5.

eqs	env
$\{(f(x,h(y),y),f(g(z),x,z))\}$	$\{\}$
$\rightarrow\{(x,g(z)),(h(y),x),(y,z)\}$	$\{\}$
$\rightarrow\{(h(y),x),(y,z)\}$	$\{x \mapsto g(z)\}$
$\rightarrow\{(x,h(y)),(y,z)\}$	$\{x \mapsto g(z)\}$
$\rightarrow\{(g(z),h(y)),(y,z)\}$	$\{x \mapsto g(z)\}$
$\rightarrow$ fail $(g \neq h)$	

Aufgabe 5. Definiere in OCaml:

```
#let unify_and_apply eqs =
let i = fullunify eqs in
let apply (t1,t2) = tsubst i t1,tsubst i t2 in
map apply eqs;;
```

Diese Funktion berechnet einen Unifikator und wendet ihn anschließend auf die ursprüngliche Liste an. Teste die Funktion an verschiedenen Instanzen. Verwende unter anderem

```
[<<|x_0|>>,<<|f(x_1,x_1)|>>;<<|x_1|>>,<<|f(x_2,x_2)|>>;
<<|x_2|>>,<<|f(x_3,x_3)|>>]
```

Interpretiere dieses Ergebnis hinsichtlich Laufzeit und Größe des Unifikators!

Lösung:
Als Ergebnis sollte erscheinen:

```
- : (term * term) list =
[(<<|f(f(f(x_3,x_3),f(x_3,x_3)),f(f(x_3,x_3),f(x_3,x_3)))|>>,
<<|f(f(f(x_3,x_3),f(x_3,x_3)),f(f(x_3,x_3),f(x_3,x_3)))|>>);
(<<|f(f(x_3,x_3),f(x_3,x_3))|>>, <<|f(f(x_3,x_3),f(x_3,x_3))|>>);
(<<|f(x_3,x_3)|>>, <<|f(x_3,x_3)|>>)]
```

Daran kann man erkennen, dass Unifikatoren exponentiell groß werden und ihre Berechnung daher ggf exponentiell viel Zeit benötigt.
Es gibt effizientere Algorithmen, aber für unsere Zwecke ist dieser Algorithmus ausreichend.

Zusatzaufgabe: Für eine gute Lösung siehe z.B Harriosn S. 156. In dieser Aufgabe ging es aber nicht darum den Code von Harrison nachzumachen. Vielmehr sollte man sich Gedanken zum *partitions-Befehl* in Ocaml machen.