

## Substitution och unifiering

- Exempel – varför behövs substitution?
- Substitution
- Unifiering
- Den mest generella unifieraren

## Resolution kräver substitution – ett enkelt exempel

Gäller  $\{(\forall x)(\text{Fågel}(x) \rightarrow \text{Vingar}(x)), \text{Fågel}(\text{Kalle\_A})\} \models \text{Vingar}(\text{Kalle\_A})?$

Normalisering ger klausulerna

1.  $\neg\text{Fågel}(x) \vee \text{Vingar}(x)$ ,
2.  $\text{Fågel}(\text{Kalle\_A})$ ,
3.  $\neg\text{Vingar}(\text{Kalle\_A})$ .

Man borde kunna använda resolution på 1 och 2 men det kräver att  $\text{Kalle\_A}$  först substitueras för  $x$ :

$$\neg\text{Fågel}(x) \vee \text{Vingar}(x) \xrightarrow{\{\text{Kalle\_A}/x\}} \neg\text{Fågel}(\text{Kalle\_A}) \vee \text{Vingar}(\text{Kalle\_A})$$

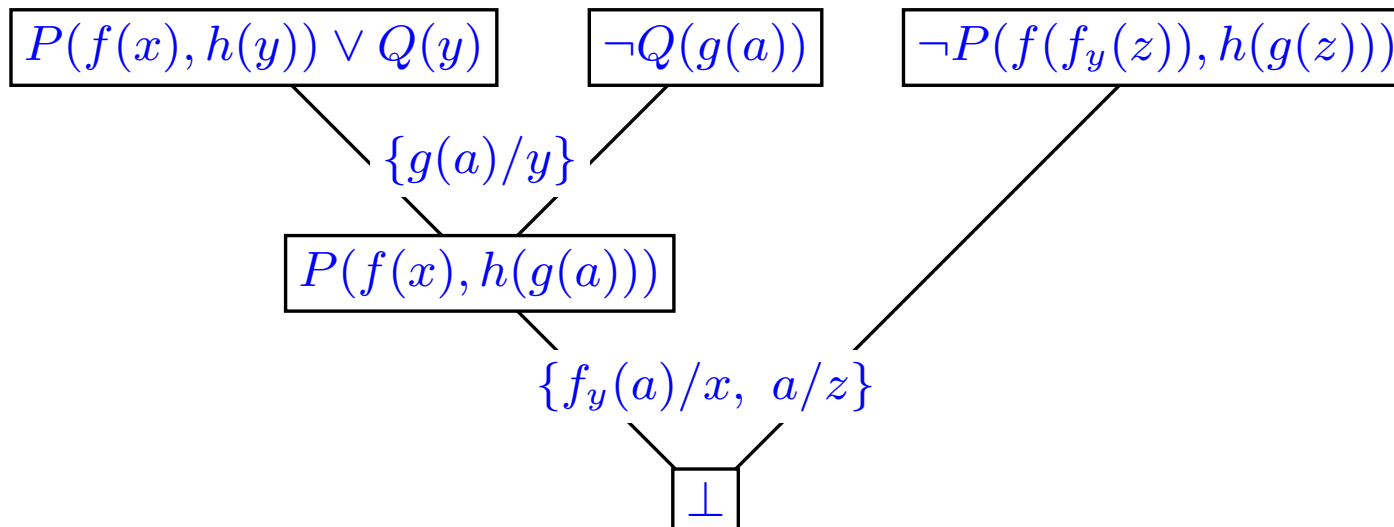
Resolution ger nu  $\text{Vingar}(\text{Kalle\_A})$  och sedan  $\perp$  pga klausul 3.

## Resolution kräver substitution – ett komplexare exempel

$$\{(\forall x)(\forall y)(P(f(x), h(y)) \vee Q(y)), (\forall x)(\neg Q(g(a)))\}$$

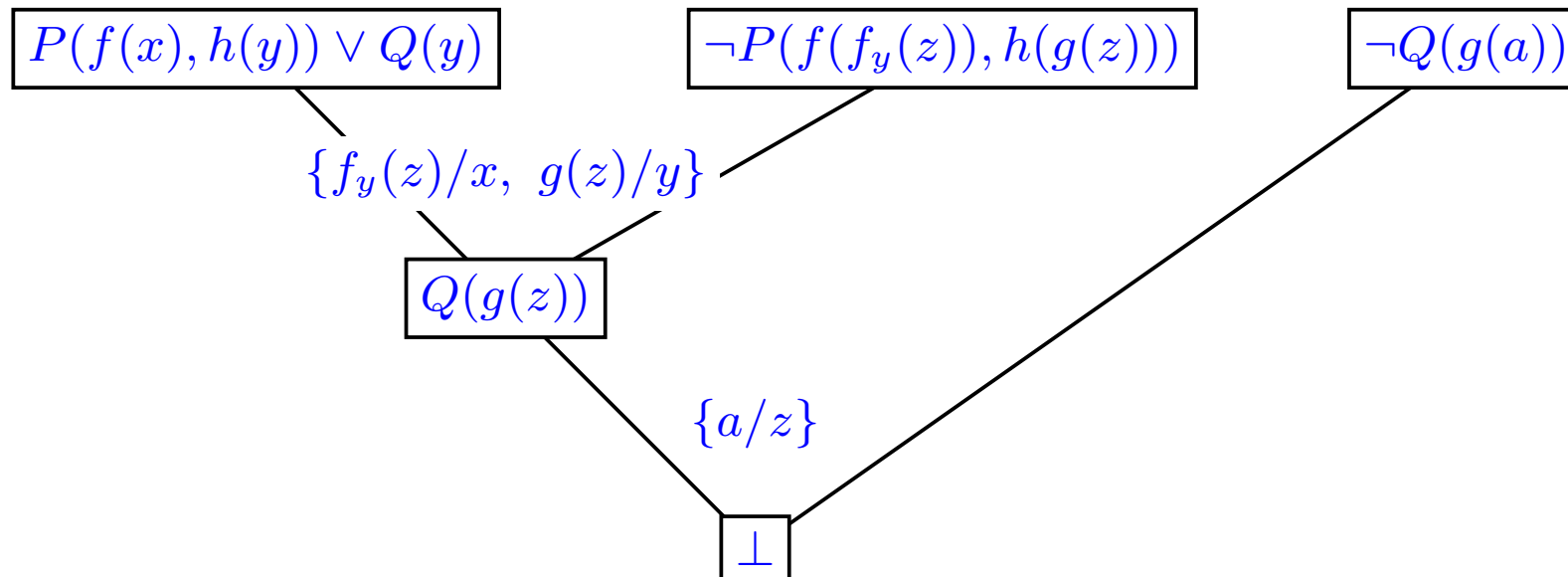
$$\models (\exists x)(\forall y)(P(f(y), h(g(x))))$$

- Normalisering ger klausulerna
1.  $P(f(x), h(y)) \vee Q(y)$ ,
  2.  $\neg Q(g(a))$ ,
  3.  $\neg P(f(f_y(z)), h(g(z)))$ .



## Samma exempel i en annan ordning

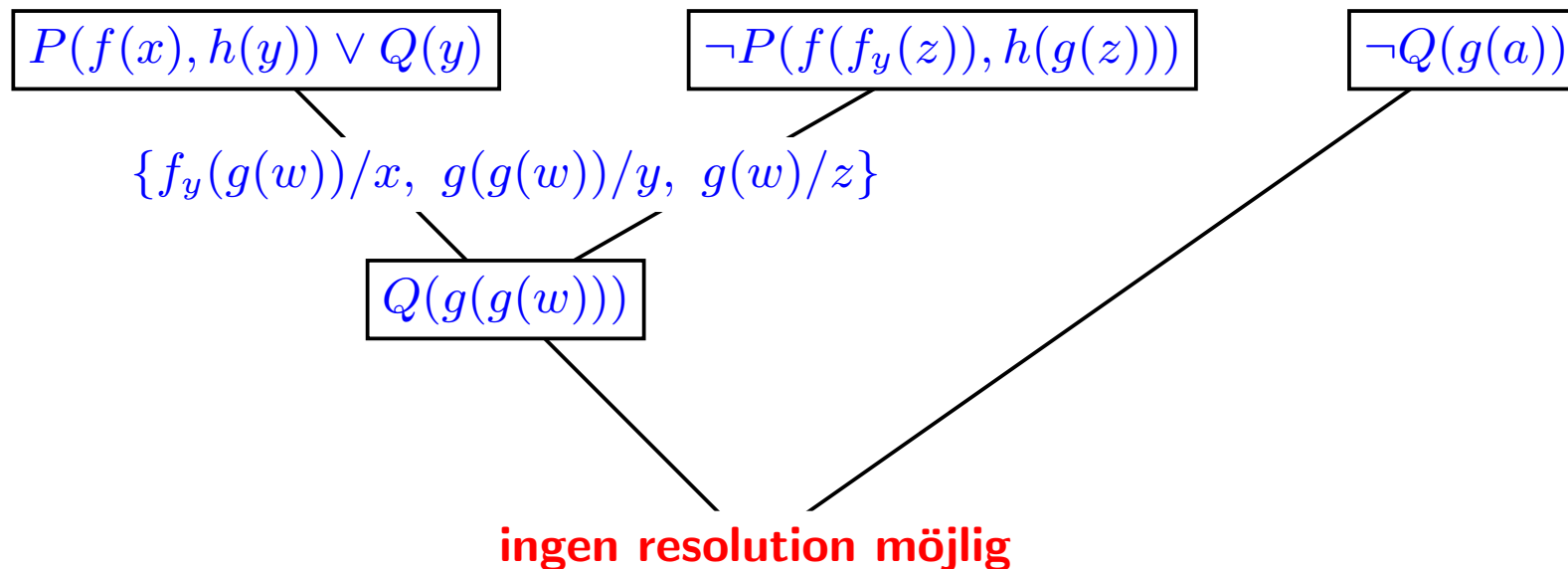
1.  $P(f(x), h(y)) \vee Q(y)$ ,
2.  $\neg Q(g(a))$ ,
3.  $\neg P(f(f_y(z)), h(g(z)))$ .



## Problem: Substitutionen får inte vara för speciell

När termer substitueras för variabler **specialiseras formeln**. Tex är  $\neg\text{Fågel}(\text{Kalle\_A}) \vee \text{Vingar}(\text{Kalle\_A})$  en specialisering av  $\neg\text{Fågel}(x) \vee \text{Vingar}(x)$ .

**Problem:** Är den speciellare än nödvändigt kan det hindra senare resolutionssteg.



## Substitution

- En **substitution** är en funktion  $\sigma: \mathbf{V} \rightarrow \text{terms}(\mathbf{T})$  där  $\sigma(x) = x$  för alla utom ändligt många  $x \in \mathbf{V}$ . Traditionellt skrivs  $x\sigma$  istället för  $\sigma(x)$ .
- En substitution är alltså en funktion på den här formen:

$$x\sigma = \begin{cases} t_1 & \text{om } x = x_1 \\ \vdots & \\ t_n & \text{om } x = x_n \\ x & \text{annars.} \end{cases}$$

Den skrivs därför vanligtvis som en mängd  $\{t_1/x_1, \dots, t_n/x_n\}$ .

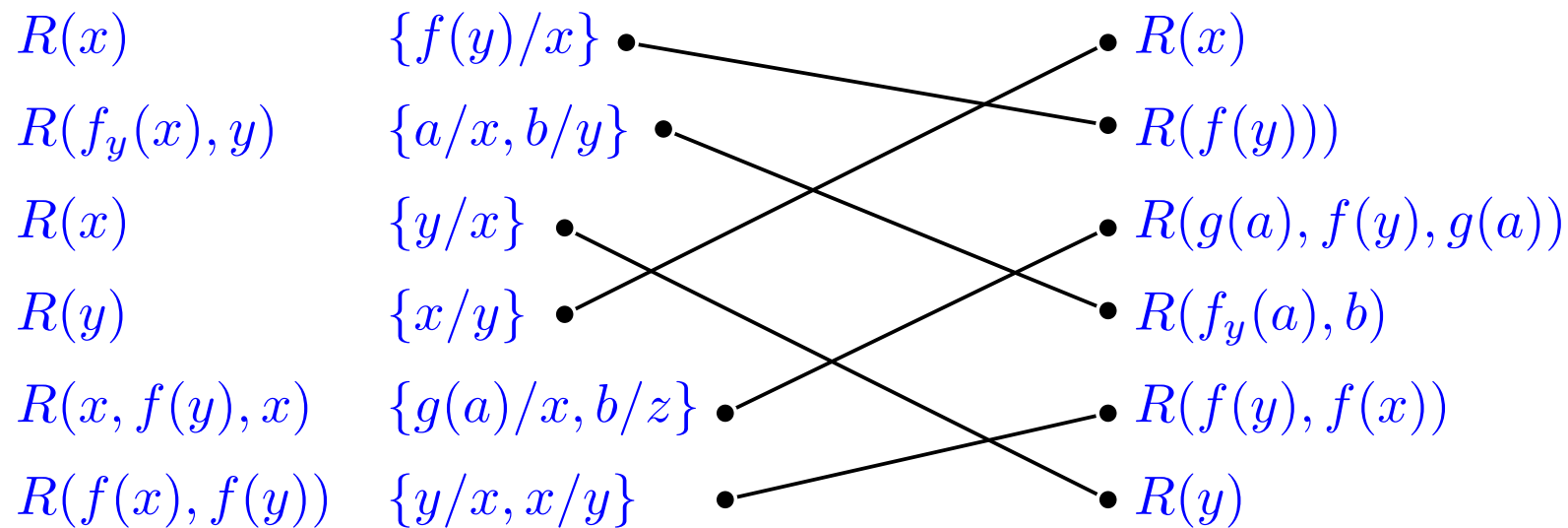
- För  $t \in \text{terms}(\mathbf{T})$  betecknar  $t\sigma$  termen som fås av  $t$  genom att simultant ersätta varje förekomst av en variabel  $x$  med  $x\sigma$ .
- Med andra ord, varje  $x_i$  ersätts med  $t_i$  ( $1 \leq i \leq n$ ) medan de övriga variablerna inte påverkas.

## Uppgift 1

Para ihop varje tuppel av en formel  $\varphi$  och en substituering  $\sigma$  med den resulterande formeln  $\varphi\sigma$ .

$R(x)$	$\{f(y)/x\}$	$R(x)$
$R(f_y(x), y)$	$\{a/x, b/y\}$	$R(f(y))$
$R(x)$	$\{y/x\}$	$R(g(a), f(y), g(a))$
$R(y)$	$\{x/y\}$	$R(f_y(a), b)$
$R(x, f(y), x)$	$\{g(a)/x, b/z\}$	$R(f(y), f(x))$
$R(f(x), f(y))$	$\{y/x, x/y\}$	$R(y)$

## Lösning 1





## Unifierare

En substitution appliceras på en klausul genom att applicera den på alla termer i klausulen. En **unifierare** för atomära formler  $\varphi, \varphi'$  är en substitution  $\sigma$  sådan att  $\varphi\sigma = \varphi'\sigma$ .

Kompositionen av substitutioner skrivs  $\sigma_1\sigma_2$  (först  $\sigma_1$  sedan  $\sigma_2$ ), alltså  $x(\sigma_1\sigma_2) = (x\sigma_1)\sigma_2$  för alla  $x \in \mathbf{V}$  (vilket ger  $t(\sigma_1\sigma_2) = (t\sigma_1)\sigma_2$  för alla termer  $t$ , så att vi istället kan skriva  $t\sigma_1\sigma_2$ ).

Exempel: Låt  $\sigma = \{f(x)/y, a/z\}$  och  $\sigma' = \{g(b)/x, b/z\}$ . Då blir

$$R(y, z)(\sigma\sigma') = (R(y, z)\sigma)\sigma' = R(f(x), a)\sigma' = R(f(g(b)), a)$$

om vi applicerar  $\sigma$  och  $\sigma'$  i tur och ordning. Men vi kan även komponera en ny substitution och applicera den direkt:  $\sigma\sigma' = \{f(g(b))/y, a/z\}$  och

$$R(y, z)\sigma\sigma' = R(f(g(b)), a) .$$

## Unifierare (2)

Exemplen visade att det behövs en unifierare av  $\varphi$  och  $\varphi'$  för att kunna köra resolution på  $\varphi, \neg\varphi'$  men unifieraren bör inte vara för speciell!

Men vad betyder ”speciell” precist?

En substitution  $\sigma_1$  är **generellare** än en substitution  $\sigma_2$  om det finns en tredje substitution  $\sigma$  sådan att  $\sigma_2 = \sigma_1\sigma$ . (OBS! Det är möjligt att  $\sigma_2$  i sin tur är generellare än  $\sigma_1$ . Då är  $\sigma$  bara ett namnbyte. ”Generellare” är alltså snarare ”minst lika generell“.)

**Exempel:** Låt  $\sigma_1 = \{f(y)/x, y/y\}$  och  $\sigma_2 = \{f(z)/x, z/y\}$  vara två substitutioner. Vi kan se att  $\sigma_1$  och  $\sigma_2$  lika generella eftersom det finns två substitutioner  $\sigma'_1 = \{z/y\}$  och  $\sigma'_2 = \{y/z\}$  så att  $\sigma_1 = \sigma_2\sigma'_2$  och  $\sigma_2 = \sigma_1\sigma'_1$ .

## Uppgift 2

Hitta en lämplig unifierare till varje formelpar.

$$R(x) \quad R(y)$$

$$R(g(a)) \quad R(x)$$

$$R(y) \quad R(y)$$

$$R(z, f(z)) \quad R(a, x)$$

$$R(g(x)) \quad R(f(y))$$

$$R(x) \quad R(f(x))$$

## Lösning 2

Det första och tredje paret har oändligt många unifierare. Det andra och fjärde paret har exakt en unifierare vardera, medan de två sista paren saknar unifierare.

$R(x)$	$R(y)$	$\{x/y\}, \{y/x\}, \{a/x, a/y\}, \dots$
$R(g(a))$	$R(x)$	$\{g(a)/x\}$
$R(y)$	$R(y)$	$\{\}, \{a/y\}, \{x/y\}, \{g(f(g(b)))/y\}, \dots$
$R(z, f(z))$	$R(a, x)$	$\{a/z, f(a)/x\}$
$R(g(x))$	$R(f(y))$	—
$R(x)$	$R(f(x))$	—

## Den mest generella unifieraren

**Definition** En unifierare  $\sigma$  av atomära formler  $\varphi, \varphi'$  är **mest generell** om den är generellare än alla övriga unifierare av  $\varphi, \varphi'$ .

**Teorem** Låt  $\varphi, \varphi'$  vara atomära formler.

1. Om både  $\sigma_1$  och  $\sigma_2$  är mest generella unifierare så är de lika upp till namnbyte, dvs det finns en bijektiv substitution  $\sigma: \mathbf{V} \rightarrow \mathbf{V}$  sådan att  $\sigma_2 = \sigma_1\sigma$  och  $\sigma_1 = \sigma_2\sigma^{-1}$ .
2. Om  $\varphi$  och  $\varphi'$  har en unifierare så har de även en mest generell unifierare (mgu). Pga del 1 är mgu:n unik upp till namnbyte.
3. Det finns en algoritm som prövar om  $\varphi, \varphi'$  har en mgu  $\sigma$  och returnerar  $\sigma$  om så är fallet.

## En effektiv algoritm för att hitta mgu:n

**Robinsons algoritm** är ett sätt att beräkna mgu:n för atomära formler  $\varphi, \varphi'$ .

**Indata:** Atomära formler  $\varphi = R(s_1, \dots, s_n)$  och  $\varphi' = R(t_1, \dots, t_n)$ .

**Algoritm:**

1.  $\sigma_{\text{mgu}} := \{\}$  (initialisering)
2. Om  $\varphi = \varphi'$  returnera  $\sigma_{\text{mgu}}$ . Annars hitta första positionen där symbolerna i  $\varphi$  och  $\varphi'$  är olika. Betrakta deltermerna som börjar vid den positionen.
  - (a) Om en av dem är en variabel  $x$  och den andra är en term  $t$  som inte innehåller  $x$ , tilldela

$$\varphi := \varphi\sigma, \varphi' := \varphi'\sigma \text{ och } \sigma_{\text{mgu}} := \sigma_{\text{mgu}}\sigma$$

där  $\sigma = \{t/x\}$  och upprepa steg 2.

- (b) Annars avbryt algoritmen – ingen unifierare finns.

## Exempel 1

Mgu:n för  $R(x, f(y, y), g(w))$  och  $R(g(z), z, g(x))$  (där  $w, x, y, z \in \mathbf{V}$ ) finns:

$s_1, \dots, s_n$	$t_1, \dots, t_n$	$\sigma_{\text{mgu}}$
$x, f(y, y), g(w)$	$g(z), z, g(x)$	
$g(z), f(y, y), g(w)$	$g(z), z, g(g(z))$	$g(z)/x$
$g(f(y, y)), f(y, y), g(w)$	$g(f(y, y)), f(y, y), g(g(f(y, y)))$	$g(f(y, y))/x,$ $f(y, y)/z$
$g(f(y, y)), f(y, y), g(g(f(y, y)))$	$g(f(y, y)), f(y, y), g(g(f(y, y)))$	$g(f(y, y))/x,$ $f(y, y)/z,$ $g(f(y, y))/w$

## Exempel 2, 3

Det finns ingen unifierare för  $R(x, x, g(y))$  och  $R(g(z), z, g(x))$

$s_1, \dots, s_n$	$t_1, \dots, t_n$	$\sigma_{\text{mgu}}$
$x, x, g(y)$	$g(z), z, g(x)$	
$g(z), g(z), g(y)$	$g(z), z, g(g(z))$	$g(z)/x$
avbrott eftersom $z$ förekommer i $g(z)$		

och inte heller någon för  $R(x, x, g(y))$  och  $R(g(z), f(a, a), g(x))$

$s_1, \dots, s_n$	$t_1, \dots, t_n$	$\sigma_{\text{mgu}}$
$x, x, g(y)$	$g(z), f(a, a), g(x)$	
$g(z), g(z), g(y)$	$g(z), f(a, a), g(g(z))$	$g(z)/x$
avbrott eftersom varken $g(z)$ eller $f(a, a)$ är en variabel		