AR for Horn clause logic

Introducing: Unification

How to deal with variables?

- Example:

$\forall p$ lot_maint(house(p)) ← big(house(p))
false ← lot_maint(house(Bos))

- We would like to conclude:

false ← groot(huis(Bos))

by means of generalized modus ponens.

Principle: use instantiations of the 2 Horn clauses, such that these DO 'match'.

More examples:

- We drop the universal quantification, since all variables are universally quantified anyway.
- Some examples using standard modus ponens:

related(x,y) ← parent(x,y)
parent(John,Mary)
→ related(John,Mary)

loves(John,x) ← related(John,x)
related(y,father(y))
→ loves(John,father(John))

Unification !!

Substitutions:

- Examples:

  θ = { x / g(z) , y / B}

  φ ={ x / h(g(A)) , y / g(A) , z / w}

- A substitution is a finite set of pairs of the form variable / term, such that all variables at the left-hand sides of the pairs are distinct.

- In our substitutions we will NOT allow that some variable that occurs left also occurs in some term at the right.

Applying substitutions:

- Examples:

  θ = { x / g(z) , y / B}

  p(x , f(y, z)) θ = p(g(z) , f(B, z))

  φ ={ x / h(g(A)) , y / g(A) , z / w}

  p(x, f(y, z)) φ = p(h(g(A)) , f(g(A) , w))

- Substitutions can be applied to simple expressions (atoms or terms), by replacing all occurrences of the left-side variables in the expression by the corresponding terms.

Remember the motivation:

- The two atoms in the clauses:

  lot_maint(house(p)) ← big(house(p))
  false ← lot_maint(house(Bos))

  must be made equal.

- We want substitutions that make atoms equal.

"Unifiers"

- Example:

S = {related(John,x), related(y, father(y))}

θ = {y / John, x / father(John)} is a unifier for S

S θ = {related(John,father(John))}

- Given a <u>set</u> of simple expressions S, we call a substitution θ a <u>unifier</u> for S if:

S θ is a singleton

One more refinement:

- For deduction step:

related(x,y) ← parent(x,y)
parent(John,z)

we have: S = {parent(x,y), parent(John,z)}

and there are several unifiers:

θ = {x / John, y / z}

φ = {x / John, y / Mary, z / Mary}

enz.

- Only the most general one, θ, allows to derive the strongest conclusion:

related(John,z)

Relation between these?

- Example: S = {parent(x,y), parent(John,z)}

θ = {x / John, y / z} → S θ = {parent(John,z)}

φ = {x / John, y / Mary, z / Mary} → S φ = {parent(John,Mary)}

There exists a third substitution:

σ = {z / Mary} with S φ = (S θ) σ

Most general unifier:

- Given a set of simple expressions S, a most general unifier θ for S is a unifier for S, such that for all other unifiers φ for S, there exists a third substitution σ such that:

$$S \varphi = (S \theta) \sigma$$

- Key-idea: create minimal instantiation changes!
- Notation: $\theta = mgu(S)$, or $\theta = mgu(A, B)$ for $S = \{A,B\}$

Generalized modus ponens for Horn clauses

- Generalized modus ponens must be further extended as:

$$\frac{\begin{array}{l} A \leftarrow B1 \wedge B2 \wedge \ldots \wedge \; Bi \wedge \; \ldots \wedge Bn \\ Bi' \leftarrow C1 \wedge C2 \wedge \ldots \wedge Cm \end{array}}{(A \leftarrow B1 \wedge B2 \wedge \ldots \wedge \; C1 \wedge C2 \wedge \ldots \wedge Cm \wedge \; \ldots \wedge Bn)\, \theta}$$

where $\theta = mgu(Bi, Bi')$

- Note: Bi and Bi' must have the same predicate.
- Correctness: due to correctness for all ground instances of this derivation.

Example: a few steps

$$\frac{\begin{array}{l} false \leftarrow lot\_maint(house(x)) \\ \qquad lot\_maint(house(y)) \leftarrow big(house(y)) \end{array}}{false \leftarrow big(house(y))}$$

- Observe: we will always provide the variables with new names in order to avoid 'accidental' clashes of names.
- Another step, much later:

$$\frac{\begin{array}{l} false \leftarrow showm(z) \wedge belg(z) \\ \qquad showm(Bos) \end{array}}{false \leftarrow belg(Bos)}$$

Backward procedure for Horn clauses

```
Goal := false ← B1 ∧ B2 ∧ … ∧ Bn ;
Repeat
    Select some Bi atom from the body of Goal
    Select some clause Bi' ← C1 ∧ C2 ∧ … ∧ Cm from
        T such that θ = mgu(Bi, Bi') exists
    Goal := false ← (B1 ∧ … ∧ Bi-1 ∧ C1 ∧ C2 ∧ … ∧ Cm
                     ∧ Bi+1 ∧ … ∧ Bn) θ
Until  Goal = false ←   or no more Selections possible
```

- Again: concrete versions of this generic scheme should allow for backtracking over previous selections,
  or they should treat the problem as a general search problem through the space of derivable goals.

The example again:

showm(Bos)
belg(Bos)

european(x) ← belg(x)
rich(x) ← showm(x) ∧ european(x)
big(house(x)) ← rich(x)
lot_maint(house(x)) ← big(house(x))
false ← lot_maint(house(x))

Goal	θ	Clause used
false ← lot_maint(house(x))	θ = { x1 / x}	lot_maint(house(x1)) ← big(house(x1))
false ← big(house(x))	θ = { x2 / x}	big(house(x2)) ← rich(x2)
false ← rich(x)	θ = { x3 / x }	rich(x3) ← showm(x3) ∧ european(x3)
false ← showm(x) ∧ european(x)	θ = { x4 / x }	european(x4) ← belg(x4)
false ← showm(x) ∧ belg(x)	θ = { x / Bos }	belg(Bos)
false ← showm(Bos)	θ = { }	showm(Bos)
false ←		

Why rename variables?

- Consider the derivation step:

```
false ← p(x)
        p(y) ← q(x,y)
false ← q(y,y)            θ = {x/y }
```

- Problem: p(y) ← q(x,y) is equivalent with p(y) ← q(z,y)
  so that alternatively we could perform the step:

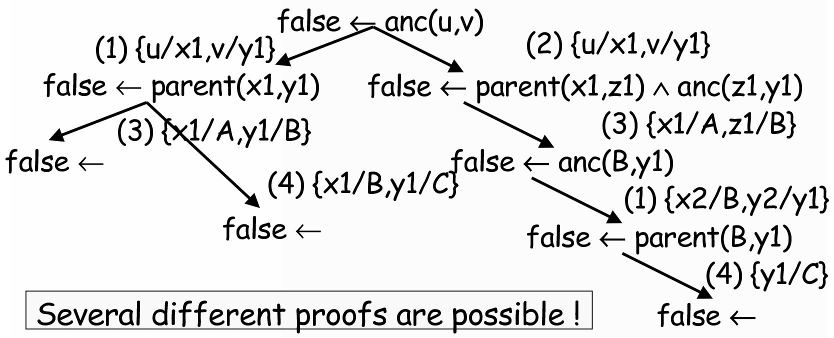
```
false ← p(x)
        p(y) ← q(z,y)
false ← q(z,y)            θ = {x/y }
```

Which gives us a strictly stronger conclusion !

Always first rename variables apart !!

Another example:

anc(x,y) ← parent(x,y) (1)
anc(x,y) ← parent(x,z) ∧ anc(z,y) (2)
parent(A,B) (3) parent(B,C) (4)
false ← anc(u,v)



Several different proofs are possible !

Completeness:

- Backward generalized modus ponens, using a complete search method to search the space of derived goals and with renaming of variables <u>is complete</u>.

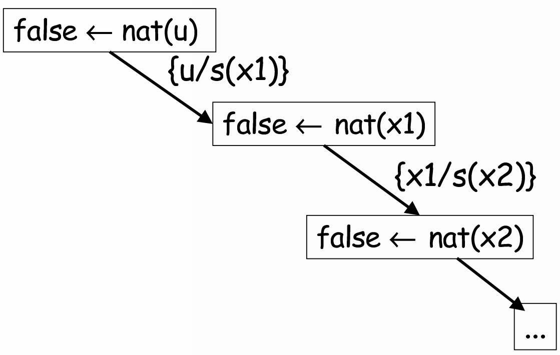
- Remark that it can <u>only be semi-deciding</u>, because the search space of goals may be infinitely large.

  thus, in general, this cannot help us to decide whether false ← is derivable.

An infinite derivation:

- <u>Example</u>:

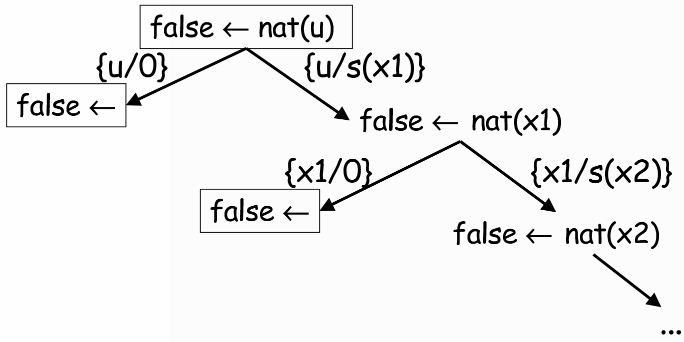
  nat(s(x)) ← nat(x)
  false ← nat(u)



Using a complete search we do get an answer for:

- Example:

```
nat(0)
nat(s(x)) ← nat(x)
false ← nat(u)
```



Unification

A basic algorithm in Automated Reasoning

A unification algorithm

```
mgu:= { s = t };
Stop:= false;
While not(Stop) and mgu still contains s = t of
   Case: t is a variable, s is not a variable:
         replace s = t by t = s in mgu;
   Case: s is a variable, t is the SAME variable:
         delete s = t from mgu;
   Case: s is a variable, t is not a variable and
          contains s :
         Stop:= true;
   ...
```

Unification algorithm (2)

...

Case: s is a variable, t is not identical to nor contains s and s occurs elsewhere in mgu:
replace all other occurrences of s in mgu by t;

Case: s is of the form f(s1,...,sn) , t of g(t1,...,tm):
if f ≠ g or n ≠ m : Stop := true;
else replace s = t in mgu by
s1 = t1 , s2 = t2 , ..., sn = tn ;

End_while

If Stop = false : Report mgu !

Example 1:

- Unify: p(B,y) and p(x,f(x)) :

  - Init: mgu:= { p(B,y) = p(x,f(x))}
  - Case 5: mgu:= {B = x, y = f(x) }
  - Case 1: mgu:= {x = B, y = f(x) }
  - Case 4: mgu:= {x = B, y = f(B) }

- No more cases applicable !
  - p(B,y) and p(x,f(x)) are unifiable
  - mgu = { x/B, y/f(B) }
  - result: p(B, f(B))

Example 2 & 3:

- Unify: p(A) and p(f(x)) :

  - Init: mgu:= { p(A) = p(f(x))}
  - Case 5: mgu:= {A = f(x) }
  - Case 5: Stop:= true → NOT unifiable!

- Unify: x and f(x) :

  - Init: mgu:= { x = f(x)}
  - Case 3: Stop:= true → NOT unifiable!

Termination of the algorithm:

- Stop = true
  - ➔ no unifier
    - ◆ expressions are not unifiable

- No more cases applicable:
  - ➔ mgu contains a set of equalities of the form:
    - ◆ {x1 = t1, …, xn = tn} with
    - ◆ all x1,…,xn mutually distinct variables !
  - ➔ The substitution {x1/t1,…,xn/tn} is a most general unifier for the initial s and t .
- Martelli-Montanari algorithm.
- Extendable for more than 2 expressions.

Deducing with unification

- Example:

vertical(segment(point(x,y),point(x,z)))
horizontal(segment(point(x,y),point(z,y)))

∃u vertical(segment(point(1,2),u))

⇨ is implied: mgu = {u/point(1,z)}

∃u,v horizontal(segment(point(1,u),point(2,v)))

⇨ is implied: mgu = {u/y,v/y }

Representation-power of Horn clauses

- Most predicate logic formulae can easily be rewritten in Horn clauses.
- Examples:

∀x cat(x) ∨ dog(x) → pet(x) ➡ pet(x) ← cat(x)
pet(x) ← dog(x)

∀x poodle(x) → dog(x) ∧ small(x) ➡ dog(x) ← poodle(x)
small(x) ← poodle(x)

- BUT:

∀x human(x) → male(x) ∨ female(x) ➡ ????

∀x dog(x) ∧ ~abnormal(x) → has_4_legs(x) ➡ ????