Predicate Logic

Using Resolution Principle for Predicate logic

To be able to use the resolution principle for predicate logic, it is important that given a formula,

- Transform it to Prenex normal form
- Skolmise the formula
- Remove the universal quantifiers
- Express the formula as a set of clauses
- Use resolution priniciple

Doctor/Quack Example

We had

F1: $\exists x(P(x) \wedge \forall y(D(y) \rightarrow l(x,y)))$

F2: $\forall x(P(x) \rightarrow \forall y(Q(y) \rightarrow \sim l(x,y)))$

To show that $\forall x(D(x) \rightarrow \sim Q(x))$

F1 can be rewritten as

$P(a) \wedge \forall y(D(y) \rightarrow l(a,y))$

$\forall y(P(a) \wedge (\sim D(y) \vee l(a,y)))$

Clauses : $P(a)$

$\sim D(y) \vee l(a,y)$

Doctor/Quack Example

F2: $\forall x(P(x) \rightarrow \forall y\big(Q(y) \rightarrow \sim l(x,y)\big))$

which can be rewritten as
$\forall x \forall y(\sim P(x) \vee \big(\sim Q(y) \vee \sim l(x,y)\big))$
$\forall x \forall y(\sim P(x) \vee \sim Q(y) \vee \sim l(x,y))$

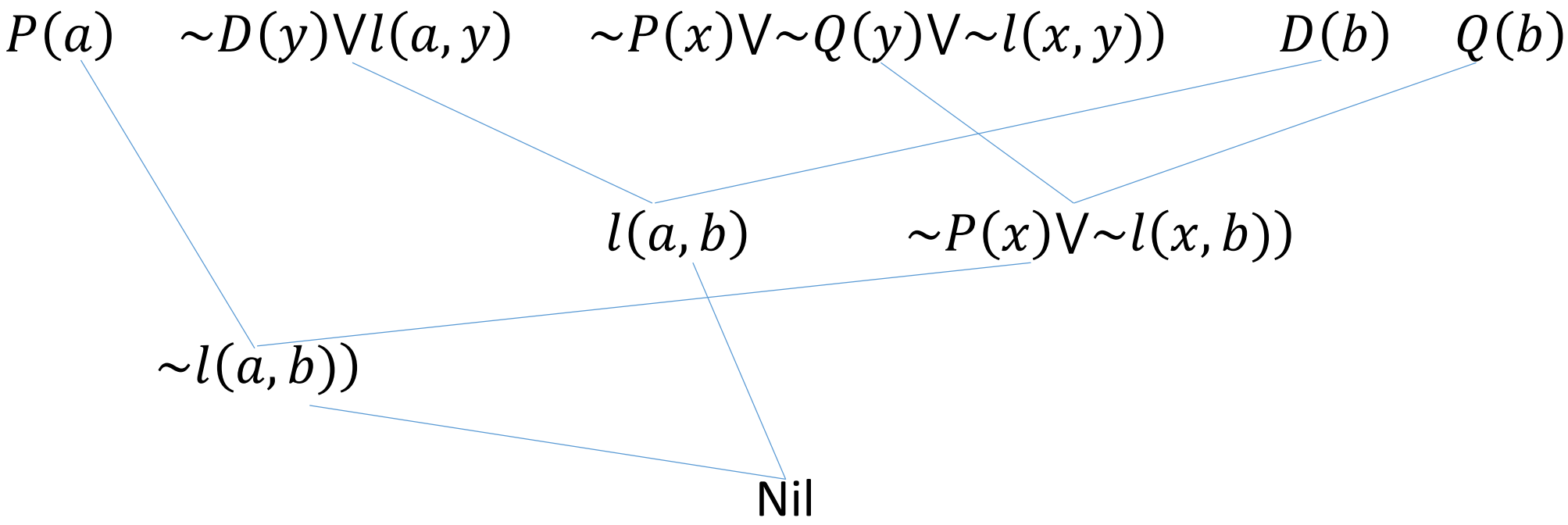
Clauses:
$\sim P(x) \vee \sim Q(y) \vee \sim l(x,y))$

To show $\forall x(D(x) \rightarrow \sim Q(x))$, it should be first negated and then brought into form of clauses.
$\sim(\forall x(D(x) \rightarrow \sim Q(x)))$
$\exists x\left(\sim\big(D(x) \rightarrow \sim Q(x)\big)\right)$

giving $\sim\big(\sim D(b) \vee \sim Q(b)\big)$ same as $D(b) \wedge Q(b)$

Clauses: $D(b)$ and $Q(b)$

Doctor/Quack Example



Hence Contradiction

Therefore $D(x) \rightarrow \sim Q(x)$ is established

The order in which clauses are combined also matter in some cases and may lead to circularity as exemplified in the next example.

Family Relationship Example

- All fathers are males.

$\forall x \forall y \; F(x, y) \rightarrow M(x)$

- If children have same father, they are siblings

$\forall x \forall y \forall w \; F(x, y) \wedge F(x, w) \rightarrow S(y, w)$

- Male sibling is brother

$\forall x \forall y \; S(x, y) \wedge M(x) \rightarrow B(x, y)$

$F(Dashrath, Ram)$

$F(Dashrath, Laxman)$

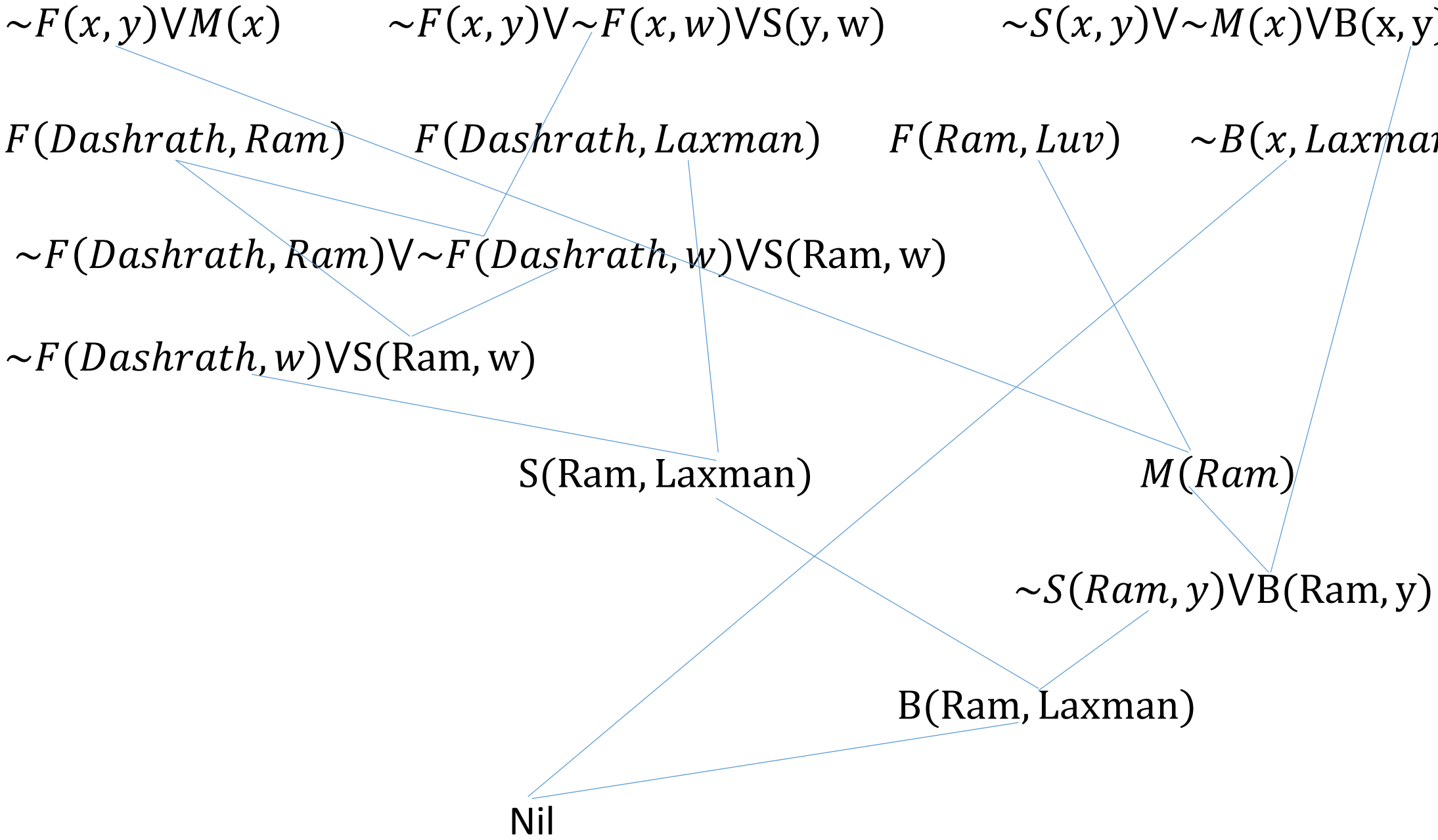
$F(Ram, Luv)$

Query: Does Laxman have a brother?

Family Relationship Example

Putting these in clausal form, we shall obtain the following clauses

- $\sim F(x, y) \mathrm{V} M(x)$
- $\sim F(x, y) \mathrm{V} \sim F(x, w) \mathrm{VS}(\mathrm{y}, \mathrm{w})$
- $\sim S(x, y) \mathrm{V} \sim M(x) \mathrm{VB}(\mathrm{x}, \mathrm{y})$
- $F(Dashrath, Ram)$
- $F(Dashrath, Laxman)$
- $F(Ram, Luv)$
- $\sim B(x, Laxman)$ (Denial of conclusion)



Hence Laxman has a brother.

In the family relationship problem, the way we proceeded ended up concluding that it was not true that Laxman has no brother. But often one needs to know

- At least one instance if the original query was satisfiable
- A mechanism to search some or all such instances that satisfy such a query

One such method is to introduce a predicate along with the negated clause which shall get bound to the constant which eliminates the negative clause.

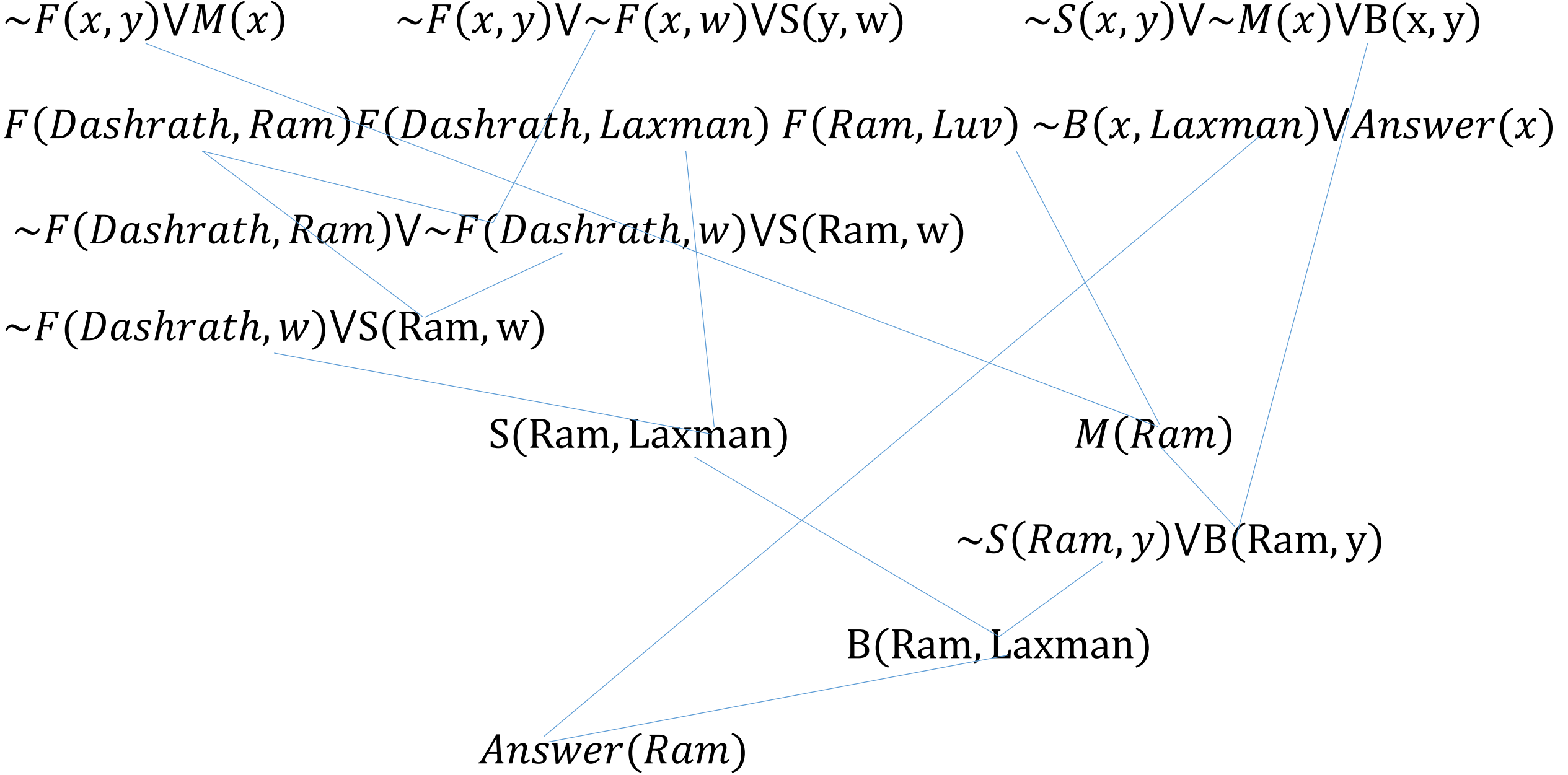
e.g., in case of family relationship example, we can write

$\sim B(x, Laxman) \vee Answer(x)$ .....(A)

instead of $\sim B(x, Laxman)$

Then, when predicate B is knocked out with a binding of Ram for x , then (A) would lead to a resolvent.

We will get Answer(Ram) instead of Nil, which will help us to answer the query.



Hence Ram is Brother of Ram.

This answered the query: Who is brother of Ram?

Typical Heuristics for generating Proofs

- Unit Preference

A clause with a single literal is called a unit clause. It is suggested that one may prefix to use unit clauses first. The reason is simple. Unit clauses lead to elimination of at least one literal thereby reducing the number of literals in the non-unit clause with which it has been resolved. Also note that two clauses with (m) and (n) (m>1, n>1) may lead to clause length upto (m+n-2).

Note: Prolog uses basically depth first strategy. Beginning with the first clause, one attempts the resolution of negated clauses.

Typical Heuristics for generating Proofs

- Set of support strategy

In family relationship example, we used the facts provided by the problem to develop the resolution argument. The clauses

- $F(Dashrath, Ram)$
- $F(Dashrath, Laxman)$
- $F(Ram, Luv)$

are usually referred to as support clauses and clauses

- All fathers are males.
- If children have same father, they are siblings
- Male sibling is brother

are regarded as axioms.

Typical Heuristics for generating Proofs

- Choose unique literals

  Choice of unique literal is advantageous in more than one way

  - Reducing the length of clauses and faster resolution
  - Less likely to lead to circularity. Clauses having multiple appearance offer no clear way to decide how to expedite resolution.

Substitution and Unification

In applying resolution principle, we have to match clauses and often obtain substitution for variables/functions that might occur.

Let us consider two clauses

$C_1: P(x) \mathrm{V} Q(x)$

$C_2: \sim P(f(x)) \mathrm{V} R(x)$

By substituting a constant ‘a’ for variable x, we obtain

$C_1^*: P(a) \mathrm{V} Q(a)$

$C_2^*: \sim P\big(f(a)\big) \mathrm{V} R(a)$

which can not be resolved.

C_1^* and C_2^* are said to be ground instances of C_1 and C_2 as no variables make their appearance in them.

Substitution and Unification

- While attempting matching for resolution, we may have several situations arising where depending on what substitution has been attempted on which variable, we may obtain different instances of the same formula.

Substitution

Let a formula F be given as $P(x, f(y), B)$

1. An instance such as $P(z, f(w), B)$ is regarded as an alphabetic variant of F.
2. $P(x, f(A), B)$ is a substitution of some variable by a constant
3. $P(g(y), f(A), B)$ is a substitution of some variable by a constant or a functional alternative.
4. $P(C, f(A), B)$ is a ground instance as there is no variable appearing in the formula.

Let us consider two clauses

$C_1: P(x) \vee Q(x)$

$C_2: \sim P(f(x)) \vee R(x)$

Substituting 'f(a)' for x in C_1 and 'a' for x in C_2, we obtain

$C_1^{**}: P(f(a)) \vee Q(f(a))$

$C_2^{**}: \sim P\big(f(a)\big) \vee R(a)$

which can be resolved.

Resolving C_1^{**} and C_2^{**}, we obtain

$C_3^{**}: Q(f(a)) \vee R(a)$

Clearly C_3^{**} is itself an instance of a general form $Q(f(x)) \vee R(x)$

Substitution

Definition

A substitution is a finite set of the form $\{t_1|v_1, t_2|v_2, \ldots, t_n|v_n\}$ where every v_i is a variable, every t_i is a term different from v_i and no two elements in the set have the same variable after the stroke symbol (' | ').

When $t_1, t_2 \ldots, t_n$ are ground terms, the substitution is called a ground substitution.

The substitution that consists of no elements is called an empty substitution and is denoted by ε.

Substitution

Let $\theta = \{t_1|v_1, t_2|v_2, \ldots, tn|vn\}$ be a substitution and E be an expression. Then $\mathrm{E}\theta$ is an expression obtained from E by replacing simultaneously each occurrence of the variable v_r, $1 \leq i \leq n$ in E by the term t_i. $\mathrm{E}\theta$ is called an instance of E.

Ex:

Let $\theta = \{a|x, f(b)|y, \ldots, c|z\}$ and $E = P(x, y, z)$

Then $\mathrm{E}\theta = P(a, f(b), c)$

Substitution

Let $\theta = \{t_1|x_1, t_2|x_2, \ldots, tn|x_n\}$

And $\lambda = \{u_1|y_1, u_2|y_2, \ldots, u_m|y_m\}$ be two substitutions.

Then composition of θ and λ is the substitution denoted by $\theta \cdot \lambda$ that is obtained from the set

$\{t_1\lambda|x_1, t_2\lambda|x_2, \ldots, t_n\lambda|x_n, u_1|y_1, u_2|y_2, \ldots, u_m|y_m\}$ by deleting any element $t_j\lambda|x_j$ for which $t_j\lambda = x_j$ and any element $u_i|yi$ such that y_i

is among $\{x_1, x_2, \ldots, x_n\}$

In simpler terms, the composition can be achieved by

- First applying substitution λ in terms of θ
- Next adding any pairs of λ having variables that do not occur in θ

Example

$\theta = \{g(x, y)|z\}$
$\lambda = \{A|x, B|y, C|w, D|z\}$

Applying substitutions of λ in terms of θ, we get
$\theta \cdot \lambda = \{g(A, B)|z\}\{A|x, B|y, C|w, D|z\}$

Next adding only the pairs of λ having variables that do not occur in θ, we obtain
$\theta \cdot \lambda = \{g(A, B)|z, A|x, B|y, C|w\}$

Example

$\theta = \{f(y)|x, z|y\}$
$\lambda = \{a|x, b|y, y|z\}$

Applying substitutions of λ in terms of θ, we get

$$\theta \cdot \lambda = \{f(b)|x, y|y\}\{a|x, b|y, y|z\}$$
$$= \{f(b)|x, y|z\}$$

Some Characteristics

- Composition of substitutions is associative.

$\theta \cdot (\lambda \cdot \mu) = (\theta \cdot \lambda) \cdot \mu$

- Empty substitution ε is both left and right identity.

$\varepsilon \cdot \theta = \theta \cdot \varepsilon$

Definition

A substitution θ is called a unifier for a set of expressions $\{E_1, E_2, \ldots, E_K\}$ if and only if $E_1\theta = E_2\theta = \cdots E_K\theta$ and the set of E's is said to be unifiable if there exist a unifier for it.

Definition

A unifier σ for a set $\{E_1, E_2, \ldots, E_K\}$ of expressions is most general unifier (m.g.u.) if and only if for each unifier θ for the set, there is a substitution λ such that $\theta = \sigma \cdot \lambda$. Further, the common instance produced by a most general unifier is unique except for alphabetic variants. (m.g.u. is a unifier with minimum number of entries).

Example

Consider a set of expressions
$\{P(x, f(y), B); P(x, f(B), B)\}$

$\theta = \{A|x, B|y\}$ produces $\{P(A, f(B), B); P(A, f(B), B)\}$

And

$\sigma = \{B|y\}$ produces $\{P(x, f(B), B); P(x, f(B), B)\}$

Note that both θ and σ are unifiers and

$\theta = \sigma.\lambda$ where $\lambda = \{A|x\}$

Note that m.g.u. is also the simplest unifier.

Unification Algorithm

- Uses a notion of disagreement set which is reduced by finding a substitution leading to a unifier.
- Disagreement Set

The disagreement set of a non empty set W of expressions is obtained by locating the first symbol (counting from the left) at which not all the expressions in W have exactly the same some symbol and the extracting from each expression in W, the subexpression that begins with the symbol occupying that position. The set of these respective subexpressions is the disagreement set of W.

Example

$W = \{P\big(x, f(y,z)\big), P(x,a), P\left(x, g\big(h, k(x)\big)\right)\}$

The disagreement set $D = \{f(y,z), a, g\big(h, k(x)\big))\}$

Unification Algorithm

1. Set $k = 0$, $W_k = W$, $\sigma_k = \varepsilon$ and unification_feasible=true
2. While unification_feasible Do
   1. If W_k is a singleton then Stop (σ_k is the m.g.u. for W)
      else find disagreement set D_k of W_k.
   2. If there does not exist any v_k and t_k in D_k such that v_k is a variable that does not occur in t_k , then Stop (Unification not feasible)
   3. Let $\sigma_{k+1} = \sigma_k\{t_k \mid v_k\}$
      and $W_{k+1} = W_k\{t_k \mid v_k\}$ (i.e. $W_{k+1} = W_k\,\sigma_{k+1}$)
   4. Set $k = k + 1$

Unification Algorithm

Set $k = 0$, $W_k = W$, $\sigma_k = \varepsilon$ and unification_feasible=true
while unification_feasible
Do
begin
If $W_k$is a singleton
then
begin
output(σ_k)
Stop
end
else

Unification Algorithm

begin

1. find disagreement set D_k of W_k
2. Find a pair v_k and t_k in D_k such that v_k is a variable not occurring in t_k
3. If such a $t_k \mid v_k$ pair exists

then

begin

Set $\sigma_{k+1} = \sigma_k\{t_k \mid v_k\}$

$W_{k+1} = W_k \sigma_{k+1}$

$k = k + 1$

end

else

begin

Output("Unification not possible")

Stop

end

end

end

Example

Find a most general unifier for
$W = \{P\left(a, x, f\left(g(y)\right)\right), P\left(z, f(z), f(u)\right)\}$

Solution:

- Set $k = 0$, $W_0 = W$, $\sigma_0 = \varepsilon$ and unification_feasible=true
- find disagreement set D_0 of W_0 $D_0 = \{a, z\}$
- In D_0, we have $v_0 = z$, $t_0 = a$ such that v_0 does not occur in t_0
- $\sigma_1 = \sigma_0\{a|z\} = \{a|z\}$

$W_1 = W_0\{t_0|\, v_0\} = W_0\{a|z\}$

$= \{P\left(a, x, f\left(g(y)\right)\right), P\left(a, f(a), f(u)\right)\}$

K=1

Example

W_1 is not a singleton

- find disagreement set D_1 $\quad D_1 = \{x, f(a)\}$
- $\sigma_2 = \sigma_1\{f(a)|x\} = \{a|z, f(a)|x\}$

$W_2 = W_1\{f(a)|x\}$

$$= \{P\left(a, f(a), f\left(g(y)\right)\right), P\left(a, f(a), f(u)\right)\}$$

K=2

Example

W_2 is not a singleton

- find disagreement set D_2 $\quad D_2 = \{g(y), u\}$
- $\sigma_3 = \sigma_2\{t_2 | v_2\} = \sigma_2\{g(y)|u\} = \{a|z, f(a)|x, g(y)|u\}$

$W_3 = W_2\{g(y)|u\}$

$$= \{P\left(a, f(a), f\left(g(y)\right)\right), P\left(a, f(a), f(g(y))\right)\}$$

$K=3$

W_3 is a singleton

So the m.g.u is $\{a|z, f(a)|x, g(y)|u\}$

- In substitution $\{t_1|v_1, t_2|v_2, \ldots, t_n|v_n\}$
  - v_i's are distinct
  - Substitution is simultaneous
- In unification algorithm
  - For the disagreement set find $t_i|\ v_i$ such that v_i does not occur in t_i

It is to be noted that while the definition of substitution would permit the use of $f(x)|x$ but unification algorithm would reject $f(x)|x$.

In the unification algorithm the effort is to find a most general unifier. It can be shown that $f(x)|x$ while a valid substitution would not lead to obtaining m.g.u. as explained on the next slide.

An explanation

- $W = \{P(x, y), P(x, z)\}$

Note that $\{y|z\}$ employs renaming to achieve unification.

- $W = \{P(x, a), P(x, z)\}$

Note that $\{a|z\}$ employs substitution of a constant for a variable.

- $W = \{P(x, f(a)), P(x, z)\}$

The substitution is $\{f(a)|z\}$

- $W = \{P(x, f(y)), P(x, z)\}$

The substitution is $\{f(y)|z\}$

All these substitutions bring these towards ground instances by restricting the universe definition by substituting for a variable.

Examples

- $W = \{P(x, a), P(x, b)\}$
  $W = \{P(x, f(a)), P(x, f(b))\}$

Disagreement set $D = \{a, b\}$ where no unification is possible as a and b are different constants.

- $W = \{P(x, f(a)), P(x, f(y))\}$

Disagreement set $D = \{a, y\}$

Unification is possible.

- $W = \{P(x, f(a)), P(x, g(y))\}$

Disagreement set $D = \{f(a), g(y)\}$

Unification is not possible as f and g are different functions.

Examples

- $W_k = \{P(x, f(y)), P(x, y)\}$

Disagreement set $D_k = \{f(y), y\}$

Clearly this is not a case of renaming of variables.

Further , on substitution

$W_{k+1} = \{P(x, f.f(y)), P(x, f(y))\}$ which is not a singleton and it alters the domain of definition of W_{k+1} from that of W_k