Horn Clauses and SLD Resolution

Marco Piastra

Horn Clauses in L_{FO}

The definition is very similar to the propositional case

- Horn Clauses (of the skolemization of a set *sentences*)

  Each clause contains at most one literal in positive form

Facts, rules and goals

Fact: a clause with just an individual *atom*

$\{Human(socrates)\}, \{Pyramid(x)\}, \{Sister(sally, motherOf(paul))\}$

Rule: a clause with at least two literals, exactly one in positive form

$\{Human(x), \neg Philosopher(x)\},$

$\forall x\ (Philospher(x) \rightarrow Human(x))$

$\{\neg Female(x), \neg Parent(k(x),x), \neg Parent(k(y),y), Sister(x,y)\}$

$\forall x \forall y\ ((Female(x) \wedge \exists z\ (Parent(z,x) \wedge Parent(z,y))) \rightarrow Sister(x,y))$

$\{\neg Above(x,y), On(x,k(x))\}, \{\neg Above(x,y), On(j(y),y)\}$

$\forall x \forall y\ (Above(x,y) \rightarrow (\exists z\ On(x,z) \wedge \exists v\ On(v,y)))$

Goal: a clause containing negative literals only

$\{\neg Human(socrates)\}$

$\{\neg Sister(sally,x), \neg Sister(x,paul)\}$

Negation of $\exists x\ (Sorella(sally,x) \wedge Sorella(x,paul))$

SLD Resolution in L_{FO}

- Input: a program Π and a goal ϕ

Program Π (i.e. a set of *definite clauses:* rules + facts) in some predefined linear order:

$\gamma_1, \gamma_2, \ldots, \gamma_n$ (each γ_i is a *definite clause*)

Goal ϕ (i.e. a conjunction of facts in negated form), which becomes the *current goal* ψ

Note: the *selection function* for the *current goal* and *subgoal* will be discussed in the next slide

Procedure:

1) Select a negative literal $\neg\alpha$ (i.e. the *subgoal*) in the *current goal* ψ
2) Scan the program (in the predefined order) to identify a clause candidate literal γ_i
3) Try unifying $\neg\alpha$ and $std(\alpha')$ (i.e. apply the standardization of variables to α')
4) If there is a *unifier* σ of $\neg\alpha$ and $std(\alpha')$, replace the current goal with the *resolvent* of $std(\gamma_i)[\sigma]$ and $\psi[\sigma]$ (i.e. first apply σ to both $std(\gamma_i)$ and ψ and then generate the resolvent)
5) Then, if the *resolvent* is the empty clause, terminate with <u>success</u>, otherwise select a new *current goal* and resume from step 1)
6) Else, if the unification fails , scan the program and select a new candidate literal γ_i and resume from step 3)
7) Else, if there are no further clauses in the program, select a new *current goal* and resume from step 1)
8) If all the goals in the tree have been fully explored, terminate with <u>failure</u>

SLD Resolution in L_{FO}

- *Two alternative selection functions:*

  Depth-first (which is the most common…)

  - Always select the *most recent goal*, i.e. the *resolvent* which has been generated last, as the *current goal* ϕ
  - Then, in the current goal ϕ, select the leftmost *subgoal* $\neg\alpha$ not selected yet
  - When the current goal ϕ is fully explored and no new *resolvent* has been generated, select the next *most recent* goal in the tree (*backtracking*)

  Breadth-first

  - Always select the *least recent* goal as the *current goal* ϕ
  - Then, in the current goal ϕ, select the leftmost *subgoal* $\neg\alpha$ not selected yet
  - When the current goal ϕ is fully explored select the next *least recent* goal in the tree

  Comparison

  Breadth-first is a *fair* selection function, in the sense that every possible resolution will be eventually attempted (i.e. 'it leaves nothing behind').

  Depth-first tends to save memory and be more efficient, but it is NOT *fair* (more to follow)

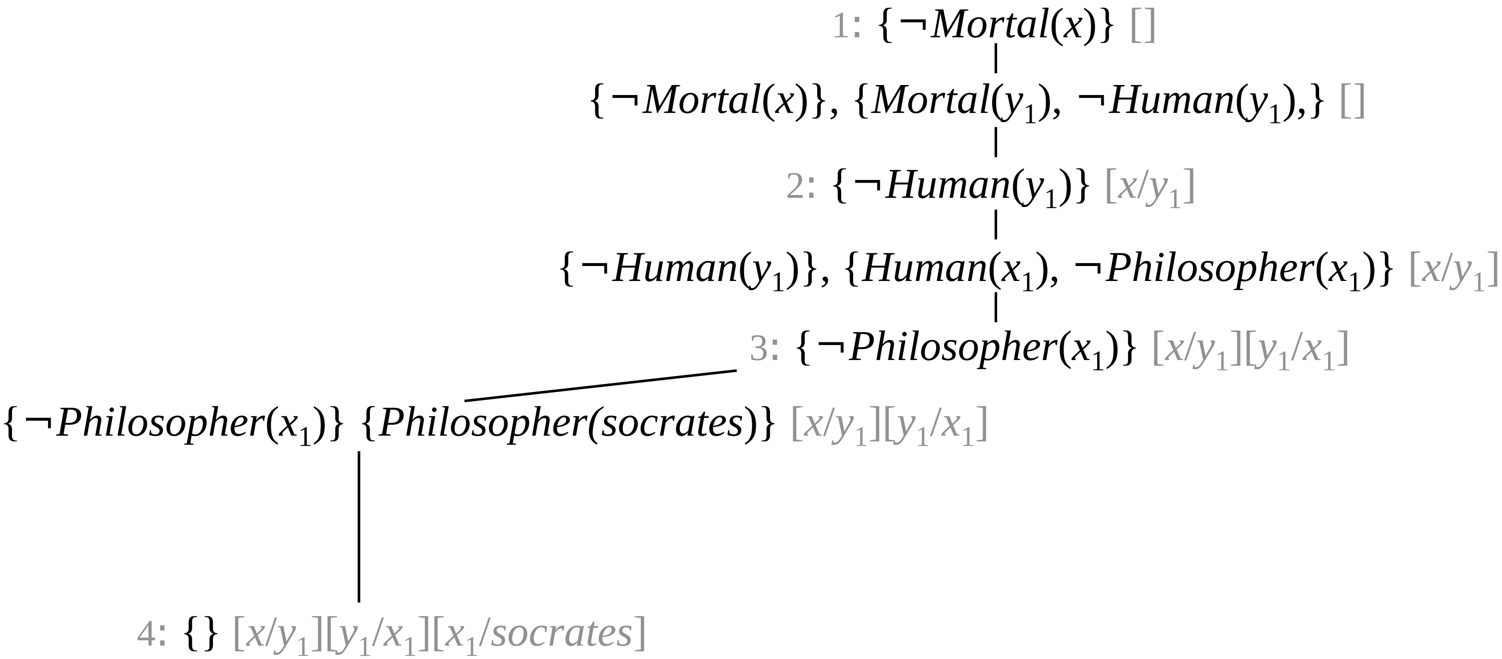
SLD Trees

- Example (depth-first selection function):

$$\Pi \equiv \{\{\mathit{Human}(x), \neg \mathit{Philosopher}(x)\}, \{\mathit{Mortal}(y), \neg \mathit{Human}(y)\},$$
$$\{\mathit{Philosopher}(\mathit{socrates})\}, \{\mathit{Philosopher}(\mathit{plato})\}, \{\mathit{Philosopher}(\mathit{aristotle})\}\}$$

$$\mathit{goal} \equiv \{\neg \mathit{Mortal}(x), \neg \mathit{Human}(x)\}$$

"Is there anyone who is both human and mortal?"



1: $\{\neg \mathit{Mortal}(x)\}$ $[]$

$\{\neg \mathit{Mortal}(x)\}, \{\mathit{Mortal}(y_1), \neg \mathit{Human}(y_1),\}$ $[]$

2: $\{\neg \mathit{Human}(y_1)\}$ $[x/y_1]$

$\{\neg \mathit{Human}(y_1)\}, \{\mathit{Human}(x_1), \neg \mathit{Philosopher}(x_1)\}$ $[x/y_1]$

3: $\{\neg \mathit{Philosopher}(x_1)\}$ $[x/y_1][y_1/x_1]$

$\{\neg \mathit{Philosopher}(x_1)\}$ $\{\mathit{Philosopher}(\mathit{socrates})\}$ $[x/y_1][y_1/x_1]$

4: $\{\}$ $[x/y_1][y_1/x_1][x_1/\mathit{socrates}]$

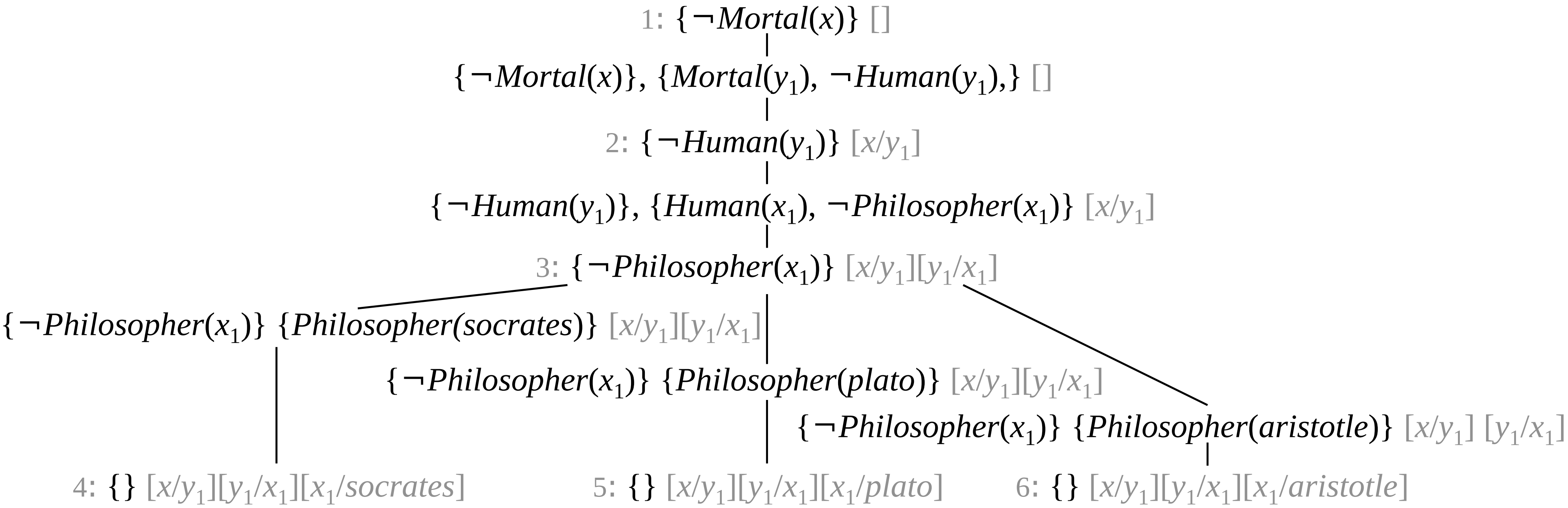
SLD Trees

- Example (depth-first selection function, forcing full exploration of SLD tree):

$$\Pi \equiv \{\{Human(x), \neg Philosopher(x)\}, \{Mortal(y), \neg Human(y)\},$$
$$\{Philosopher(socrates)\}, \{Philosopher(plato)\}, \{Philosopher(aristotle)\}\}$$

$$goal \equiv \{\neg Mortal(x), \neg Human(x)\}$$

"Is there anyone who is both human and mortal?"



1: $\{\neg Mortal(x)\}$ $[]$

$\{\neg Mortal(x)\}, \{Mortal(y_1), \neg Human(y_1),\}$ $[]$

2: $\{\neg Human(y_1)\}$ $[x/y_1]$

$\{\neg Human(y_1)\}, \{Human(x_1), \neg Philosopher(x_1)\}$ $[x/y_1]$

3: $\{\neg Philosopher(x_1)\}$ $[x/y_1][y_1/x_1]$

$\{\neg Philosopher(x_1)\}$ $\{Philosopher(socrates)\}$ $[x/y_1][y_1/x_1]$

$\{\neg Philosopher(x_1)\}$ $\{Philosopher(plato)\}$ $[x/y_1][y_1/x_1]$

$\{\neg Philosopher(x_1)\}$ $\{Philosopher(aristotle)\}$ $[x/y_1]$ $[y_1/x_1]$

4: $\{\}$ $[x/y_1][y_1/x_1][x_1/socrates]$

5: $\{\}$ $[x/y_1][y_1/x_1][x_1/plato]$

6: $\{\}$ $[x/y_1][y_1/x_1][x_1/aristotle]$

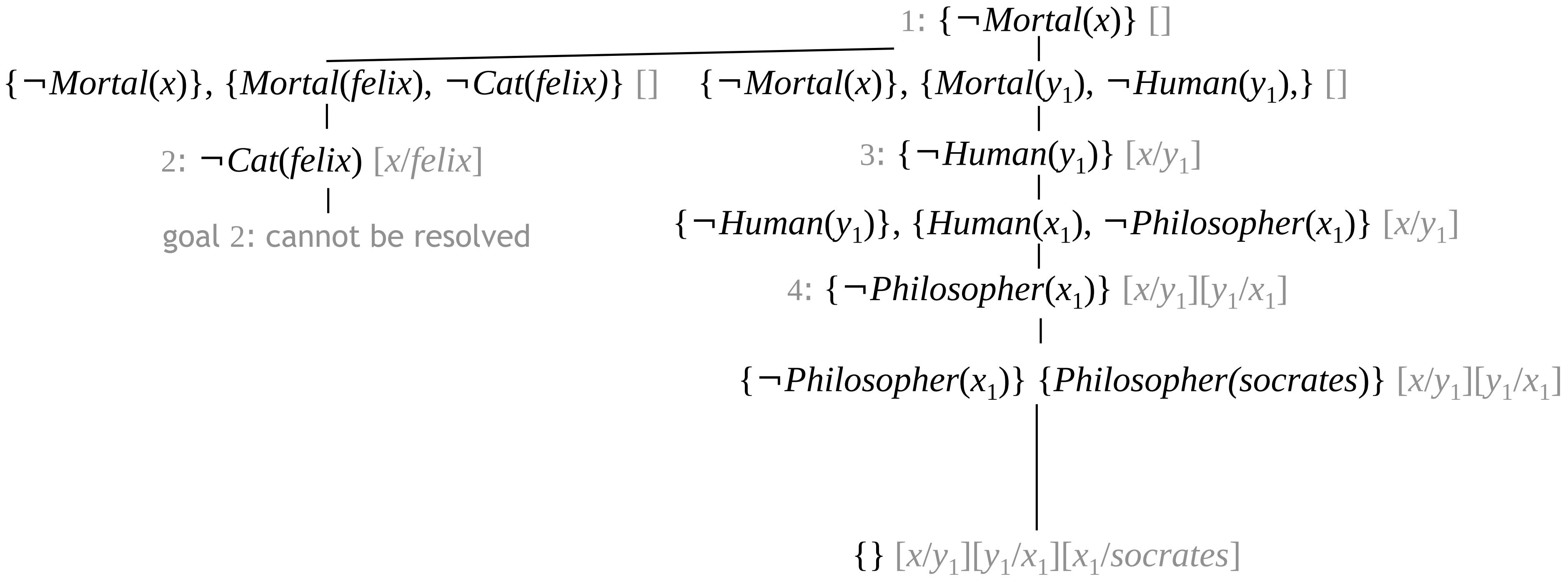
SLD Trees

- Another example (depth-first selection function):

$$\Pi \equiv \{\{Mortal(felix), \neg Cat(felix)\}, \{Human(x), \neg Philosopher(x)\}, \{Mortal(y), \neg Human(y)\},$$
$$\{Philosopher(socrates)\}, \{Philosopher(plato)\}, \{Philosopher(aristotle)\}\}$$

$$goal \equiv \{\neg Mortal(x), \neg Human(x)\}$$

"Is there anyone who is both human and mortal?"



1: $\{\neg Mortal(x)\}$ $[]$

$\{\neg Mortal(x)\}, \{Mortal(felix), \neg Cat(felix)\}$ $[]$

2: $\neg Cat(felix)$ $[x/felix]$

goal 2: cannot be resolved

$\{\neg Mortal(x)\}, \{Mortal(y_1), \neg Human(y_1),\}$ $[]$

3: $\{\neg Human(y_1)\}$ $[x/y_1]$

$\{\neg Human(y_1)\}, \{Human(x_1), \neg Philosopher(x_1)\}$ $[x/y_1]$

4: $\{\neg Philosopher(x_1)\}$ $[x/y_1][y_1/x_1]$

$\{\neg Philosopher(x_1)\}$ $\{Philosopher(socrates)\}$ $[x/y_1][y_1/x_1]$

$\{\}$ $[x/y_1][y_1/x_1][x_1/socrates]$

The world of lists

- Lists of items $[a, b, c, \ldots]$

  $cons/2$
  it's a function that associates items (e.g. a) to a list (e.g. $[b, c]$)
  $cons(a,cons(b,cons(c,nil)))$ is the list $[a, b, c]$

  $Append/3$
  it's a predicate: each pair of lists x and y is associated to their *concatenation* z

  nil
  it's a constant, the *empty list*.

Shorthand notation (Prolog):

$[] \Leftrightarrow nil$
$[a] \Leftrightarrow cons(a,nil)$
$[a,b] \Leftrightarrow cons(a,cons(b,nil))$
$[a|[b,c]] \Leftrightarrow cons(a,[b,c])$

Axioms ($\mathbf{AL}$)

$\forall x\, Append(nil,x,x)$
$\forall x \forall y \forall z\, (Append(x,y,z) \rightarrow \forall s\, Append([s,x],y,[s,z]))$

The world of lists

Problem: $\forall x\ Append(nil, x, x) \models \exists y\ \forall x\ Append(nil, cons(y, x), cons(a, x))$

1: $\forall x\ Append(nil, x, x),\ \neg\exists y\ \forall x\ Append(nil, cons(y, x), cons(a, x))$ (refutation)

2: $\forall x\ Append(nil, x, x),\ \forall y\ \exists x\ \neg Append(nil, cons(y, x), cons(a, x))$ (prenex normal form)

3: $\{Append(nil, x, x)\}, \{\neg Append(nil, cons(y, k(y)), cons(a, k(y)))\}$

($k/1$ is a Skolem function, clausal form)

(N.B. there is no *skolemization* in Prolog : the programmer does it)

The pair of **literals**

$Append(nil, x, x),\ \neg Append(nil, cons(y, k(y)), cons(a, k(y))))$

… contains the same predicate $Append/3$ but the arguments are <u>**different**</u>

There is however an MGU $\sigma = [x/cons(a, k(a)), y/a]$ that yields

$\{Append(nil, cons(a,k(a)), cons(a,k(a)))\}, \{\neg Append(nil, cons(a, k(a)), cons(a, k(a)))\}$

From this, the resolvent is the empty clause.

The world of lists in Prolog

```
% Identical to built-in predicate append/3, although it uses "cons"
% as a defined predicate, thus allowing trace-ability.

append(cons(S,X),Y,cons(S,Z)) :- append(X,Y,Z).
append(nil,X,X).

% WARNING: express your queries with cons. Examples:
% ?- append(cons(a,nil), cons(b,cons(c, nil)),cons(a,cons(b,cons(c, nil)))).
% ?- append(X,Y,cons(a,cons(b,cons(c, nil)))).
```

Infinite SLD Trees (*fairness of SLD*)

- A first example:

$\Pi \equiv \{\{P(x), \neg P(x)\}\}$

$\neg\phi \equiv \{\neg P(x)\}$

goal: $\neg P(x)$ []

$\{\neg P(x)\}, \{P(x_1), \neg P(x_1),\}$ []

$\{\neg P(x_1)\}$ $[x/x_1]$

$\{\neg P(x_1)\}, \{P(x_2), \neg P(x_2),\}$ $[x/x_1]$

$\{\neg P(x_2)\}$ $[x/x_1]$ $[x_1/x_2]$

…

Since $\Pi \not\models \phi$, the method can *diverge*

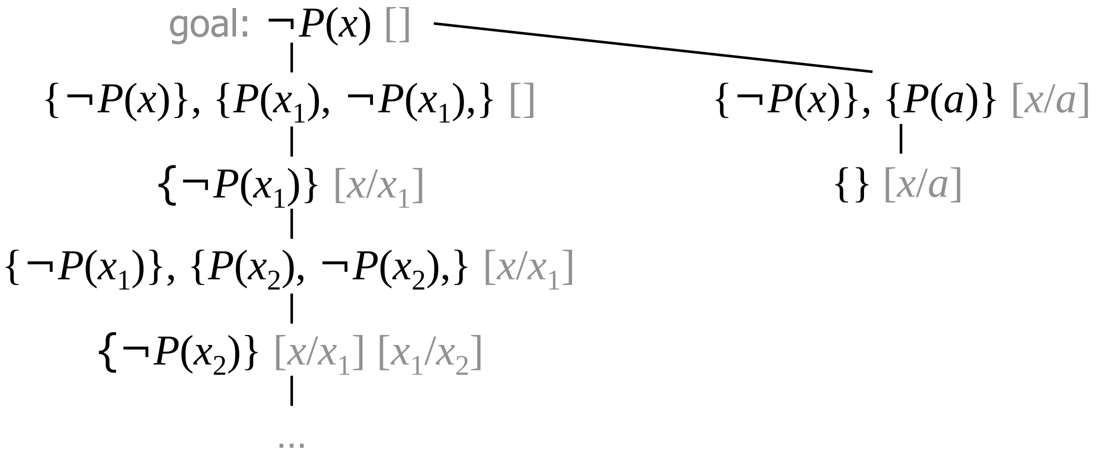
(*although a divergence of this kind can be easily spotted and avoided …*)

Infinite SLD Trees (*fairness of SLD*)

- A second example:

$\Pi \equiv \{\{P(x), \neg P(x)\}, \{P(a)\}\}$

$\neg\phi \equiv \{\neg P(x)\}$



goal: $\neg P(x)$ $[]$

$\{\neg P(x)\}, \{P(x_1), \neg P(x_1),\}$ $[]$

$\{\neg P(x_1)\}$ $[x/x_1]$

$\{\neg P(x_1)\}, \{P(x_2), \neg P(x_2),\}$ $[x/x_1]$

$\{\neg P(x_2)\}$ $[x/x_1]$ $[x_1/x_2]$

…

$\{\neg P(x)\}, \{P(a)\}$ $[x/a]$

$\{\}$ $[x/a]$

In this case $\Pi \models \phi$, so the method should *not* diverge.

However, when a *depth-first* selection function is used, the infinite branch in the SLD-tree makes the method diverge anyway.

A ***fair*** selection function is such that no possible resolution will be postponed indefinitely: that is, any possible resolution will be performed, eventually.