Predicate logic

Language

- constants
- variables
- connectives
- quantifiers – universal $\forall$, existential $\exists$
- predicate symbols – predicate = n-ary relation
- function symbols
- punctuation

Formulas

- terms = constants, variables, $f(t_1, ..., t_n)$
  ground terms = variable–free terms
- atomic formula $R(t_1, ..., t_n)$, arity, arguments
- formulas
  - atomic formulas
  - $\neg F$, $F\ OP\ G$ (OP = $\wedge, \vee, \rightarrow$.)
  - $\exists F, \forall F$
- sentence = no free occurence of any variable (all variables are bound)
- open formula = without quantifiers

Substitution

- only free variables
- If the term t contains an occurence of some variable x (which is necessarily free in t) we say that t is *substitutable* for the free variable v in the formula $A(v)$ if all occurencies of x in t remains free in $A(v/t)$
- Example:$A = \exists x P(x, y)$
  $A(y/z) = \exists x P(x, z)$
  $A(y/2) = \exists x P(x, 2)$
  $A(y/f(z, z)) = \exists x P(x, f(z, z))$.
  but not
  $A(y/f(x, x)) = \exists x P(x, f(x, x))$

Axiomatic system for predicate calculus

- axioms (A, B, C = formulas):

  $\mathbf{A_1}$ $A \Rightarrow (B \Rightarrow A)$

  $\mathbf{A_2}$ $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

  $\mathbf{A_3}$ $(\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$

  $\mathbf{A_4}$ $(\forall x)\alpha(x) \rightarrow \alpha(t)$ for any term that is substitutable for x in α

  $\mathbf{A_5}$ $(\forall x)(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow (\forall x)\beta)$ if α contains no occurence of x

- two inference rules

  Modus Ponens

$$\frac{A \qquad A \Rightarrow B}{B}$$

Generalization From $\forall x \alpha$ infer α

- proof = a finite sequence of formulas

Prenex normal forms

- DNF, CNF

$$Qx_1 \ldots Qx_n((A_{1_1} \vee \ldots \vee A_{1_{l_1}}) \wedge (A_{2_1} \vee \ldots \vee A_{2_{l_2}}) \wedge \ldots \wedge (A_{m_1} \vee \ldots \vee A_{m_{l_m}}))$$

- Example:

$$\forall x \forall y \exists z \forall w((P(x,y) \vee \neg Q(z)) \wedge (R(x,w) \vee R(y,w)))$$

- Every formula ϕ has a prenex equivalent.

Algorithm

1. Remove the quantifiers that are not used
2. Rename variables so that each quantifier has a unique variable
3. Eliminate all connectives but $\neg, \wedge$ a $\vee$
4. Move negation to the right
   $\neg\forall x A$ --> $\exists x \neg A$
   $\neg(A \wedge B)$ --> $\neg A \vee \neg B$ apod.
5. Move quantifiers to the left ($\mathtt{op} \in \{\wedge, \vee\}, Q \in \{\forall, \exists\}$):
   $A \ \mathtt{op}\ QxB$ --> $Qx(A \ \mathtt{op}\ B)$
   $QxA \ \mathtt{op}\ B$ --> $Qx(A \ \mathtt{op}\ B)$

6. Use distributive laws

$A \vee (B \wedge C)$ --> $(A \vee B) \wedge (A \vee C)$

$(A \wedge B) \vee C$ --> $(A \vee C) \wedge (B \vee C)$

Skolemization

- Skolem Normal Form - NF with universal quantifiers
- $\forall x_1 \ldots \forall x_n \exists y P(x_1, \ldots, x_n, y)$
  $-->$
  $\forall x_1 \ldots \forall x_n P(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$
- Example:

  $\forall x \exists y (x + y = 0) --> \forall x (x + f(x) = 0)$

  For the domain of integers with the operation $+$: f = inverse number,
- not equivalent but equisatisfiable

Skolemization: Algorithm

1. transform the formula into NF
2. replace all existentially quantified variables with Skolem functions.

   Arguments of SF = all universally quantified vars that have appeared before the variable.

- Example 2: $\forall x \exists y \neg (P(x, y) \Rightarrow \forall z R(y)) \vee \neg \exists x Q(x)$

  1. $\forall x_1 \exists y \forall x_2 ((P(x_1, y) \vee \neg Q(x_2)) \wedge (\neg R(y) \vee \neg Q(x_2)))$
  2. $\forall x_1 \forall x_2 ((P(x_1, f(x_1)) \vee \neg Q(x_2)) \wedge (\neg R(f(x_1)) \vee \neg Q(x_2)))$

- Example 3: $\forall x \exists y \forall z \exists w (P(x, y) \vee \neg Q(z, w))$

$$\forall x \forall z (P(x, f_1(x)) \vee \neg Q(z, f_2(x, z)))$$

Herbrand's Theorem I

- looking for the simplest interpretation; Skolem normal form, all the constants (maybe +1), functions and predicate symbols
- *Herbrand universe* $U(S)$ = all such terms

  Example:

  For $S = \{P(f(0))\}$,

  $U(S) = \{0, f(0), f(f(0)), f(f(f(0))), \ldots\}$
- *Herbrand base* $B(S)$ = all atomic formulas build upon $U(S)$;

  $B(S) = \{P(t_1, \ldots, t_n) | t_i \in$
  $U(S), P$ ... a predicate symbol from$S\}$

  Example:

  For $S = \{P(f(0))\}$,

  $B(S) = \{P(0), P(f(0)), P(f(f(0))), \ldots\}$

Herbrand's Theorem II

- *Herbrand structure* (in Czech interpretace) is a subset of $B(S)$.
- *Herbrand model* $M(S)$ of S is an Herbrand structure which is model of S, i.e. every sentence f S s true in $M(S)$.
- **Herbrand's Theorem:** Let S be a set of open formulas of a language L. Either
  1. S has an Herbrand model or
  2. S is unsatifiable and, in particular, there are finitely many ground instances of elements of S whose conjunction is unsatisfiable.

Consequence: we do not need to explore any other structures but Herbrand

Resolution in predicate logic – introduction

- based on *refutation*
- suitable for automated theorem proving
- formulas in Skolem normal form
  - clause = disjunction of literals (atoms or negation of atoms), represented as a set
  - formula = conjunction of clauses, represented as a set
- Example:
  $\forall x \forall y ((P(x, f(x)) \vee \neg Q(y)) \wedge (\neg R(f(x)) \vee \neg Q(y)))$
  $\longrightarrow$
  $\{\{P(x, f(x)), \neg Q(y)\}, \{\neg R(f(x)), \neg Q(y)\}\}$

Unification

- a substitution ϕ is a *unifier* for $S = \{E_1, \ldots, E_2\}$ if $E_1\phi = E_2\phi = \ldots = E_n\phi$, i.e., $S\phi$ is singleton.

  S is said to be *unifiable* if it has a unifier.

- a unifier ϕ for S is a *most general unifier (mgu) for* S if, for every unifier ψ for S, there is a substitution λ such that $\phi\lambda = \psi$

  up to renaming variables there is only one result applying an mgu

Unification – Examples

1. a unifier for $\{P(x,c), P(b,c)\}$ is
   $\phi = \{x/b\}$; is there any other?

2. a unifier for $\{P(f(x),y), P(f(a),w)\}$ is
   $\phi = \{x/a, y/w\}$
   but also $\psi = \{x/a, y/a, w/a\}$,
   $\sigma = \{x/a, y/b, w/b\}$ etc.

3. $\{P(x,a), P(b,c)\}, \{P(f(x),z), P(a,w)\}$,
   $\{P(x,w), \neg P(a,w)\}$,
   $\{P(x,y,z), P(a,b)\}, \{R(x), P(x)\}$
   are not unifiable

mgu?

in (2.) ϕ is the mgu: $\psi = \phi\{w/a\}, \sigma = \phi\{w/b\}$

Resolution in predicate logic – preliminaries

- variables are local for a clause (pozn.: $\forall x(A(x) \wedge B(x)) \Leftrightarrow (\forall x A(x) \wedge \forall x B(x)) \Leftrightarrow (\forall x A(x) \wedge \forall y B(y))$)

  i.e. there is no relation between variables equally named

- standardization of vars = renaming, necessary

  $\{\{P(x)\}, \{\neg P(f(x))\}\}$ is unsatisfiable. Without renaming a variable no unification can be performed

Resolvent – Examples

Example 1: $\{P(x, a)\}$, $\{\neg P(x, x)\}$

- rename vars: $\{P(x_1, a)\}$
- $mgu(\{P(x_1, a), P(x, x)\}) = \{x_1/a, x/a\}$
- resolvent $\Box$

Example 2: $\{P(x, y), \neg R(x)\}$, $\{\neg P(a, b)\}$

- $mgu(\{P(x, y), P(a, b)\}) = \{x/a, y/b\}$
- apply mgu to $\{\neg R(x)\}$
- resolvent $\{\neg R(a)\}$

Resolution rule in predicate logic

C_1, C_2 clauses that have no variables in common in the form

$$C_1 = C_1' \sqcup \{P(\vec{x}_1), \ldots, P(\vec{x}_n)\},$$
$$C_2 = C_2' \sqcup \{\neg P(\vec{y}_1), \ldots, \neg P(\vec{y}_m)\}$$

respectively. If ϕ is an mgu for $\{P(\vec{x}_1), \ldots, P(\vec{x}_n), P(\vec{y}_1), \ldots, P(\vec{y}_m)\}$, then $C_1'\phi \cup C_2'\phi$ is a *resolvent* of C_1 and C_2 (also called the *child* of *parents* C_1 and C_2).

Resolution rule in predicate logic II

- *Resolution proofs of C from S* is a finite sequence $C_1, C_2, ..., C_N = C$ of clauses such that each C_i is either a member of S or a resolvent of clauses C_j, C_k for $j, k < i$
- *resolution tree proof* C from S is a labeled binary tree
  the root is labeled C
  the leaves are labeled with elements of S and
  if any nonleaf node is labeled with C_2 and its immediate successors are labeled with C_0, C_1 then C_2 is a resolvent C_0 and C_1
- *(resolution) refutation* of S is a deduction of $\Box$ from S

Resolution – Examples II

Ex. 3: $C_1 = \{Q(x), \neg R(y), P(x,y), P(f(z), f(z))\}$ a
$C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}$

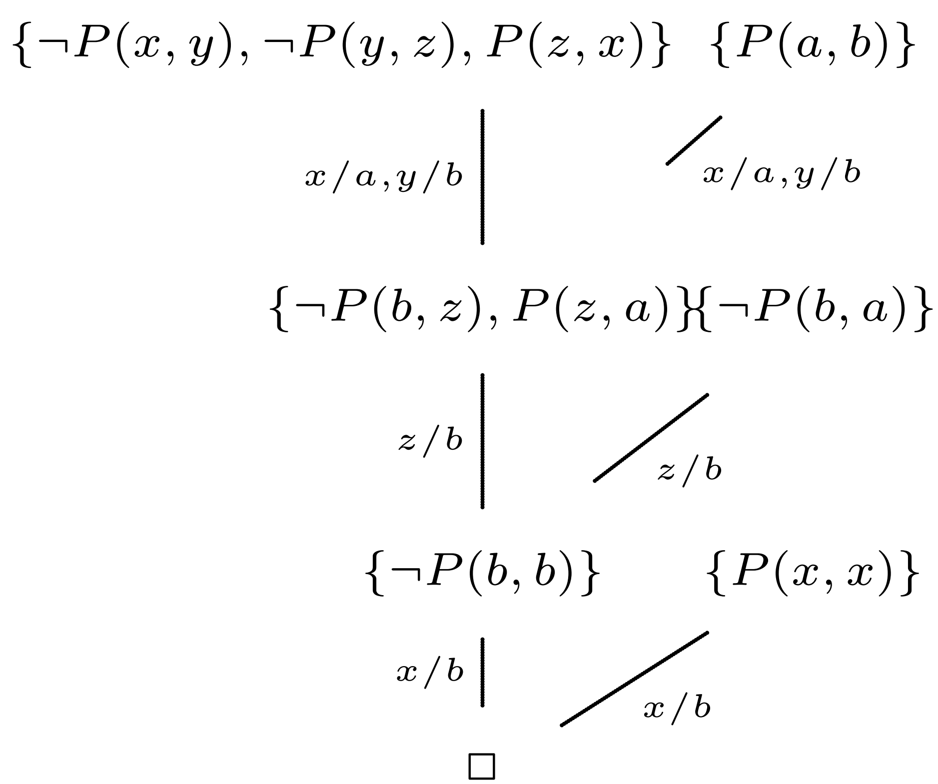
- choose the set of literal
  $\{P(x,y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$
- mgu $\phi = \{x/f(a), y/f(a), z/a, w/a\}$
- $C_1' = \{Q(x), \neg R(y)\}, C_1'\phi = \{Q(f(a)), \neg R(f(a))\}$
- $C_2' = \{\neg N(u), \neg R(w)\}, C_2'\phi = \{\neg N(u), \neg R(a)\}$
- the resolvent
  $C_1'\phi \cup C_2'\phi = \{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}$

Resolution in the predicate logic

- is sound (soundness) and complete
- systematic attempts at generating resolution proofs possible but redundant and inefficient: the search space is too huge
- what strategy of generating resolvents to choose?

Linear resolution

$$\{\{P(x,x)\},\{\neg P(x,y),\neg P(y,z),P(z,x)\},\{P(a,b)\},\{\neg P(b,a)\}\}$$

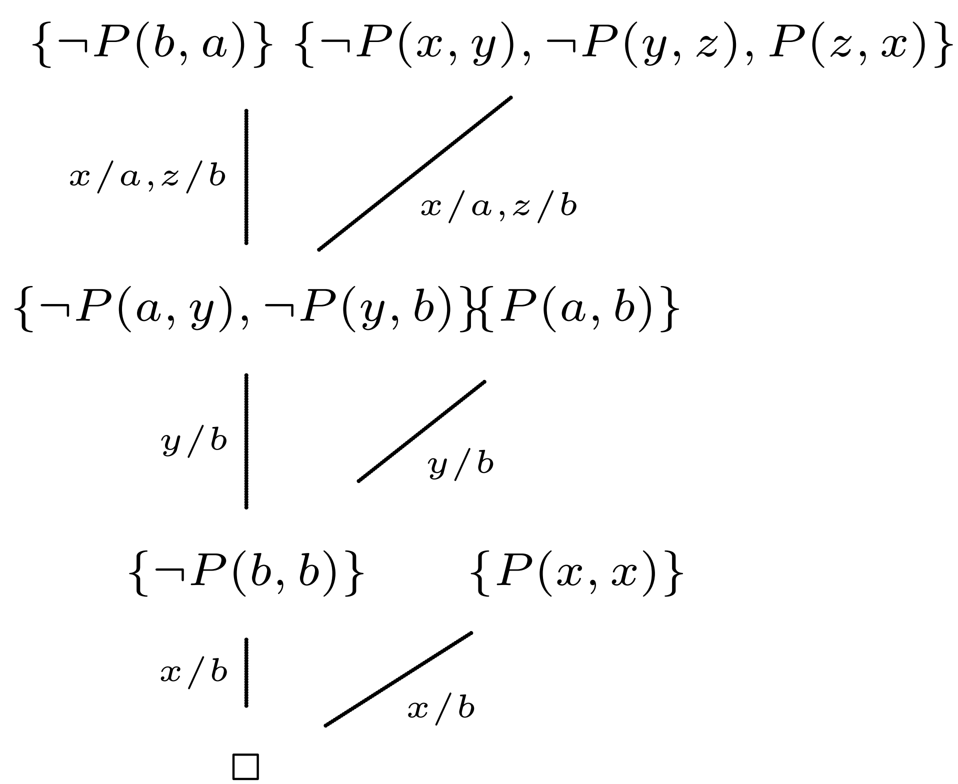


sound and complete

LI-resolution

linear input resolution

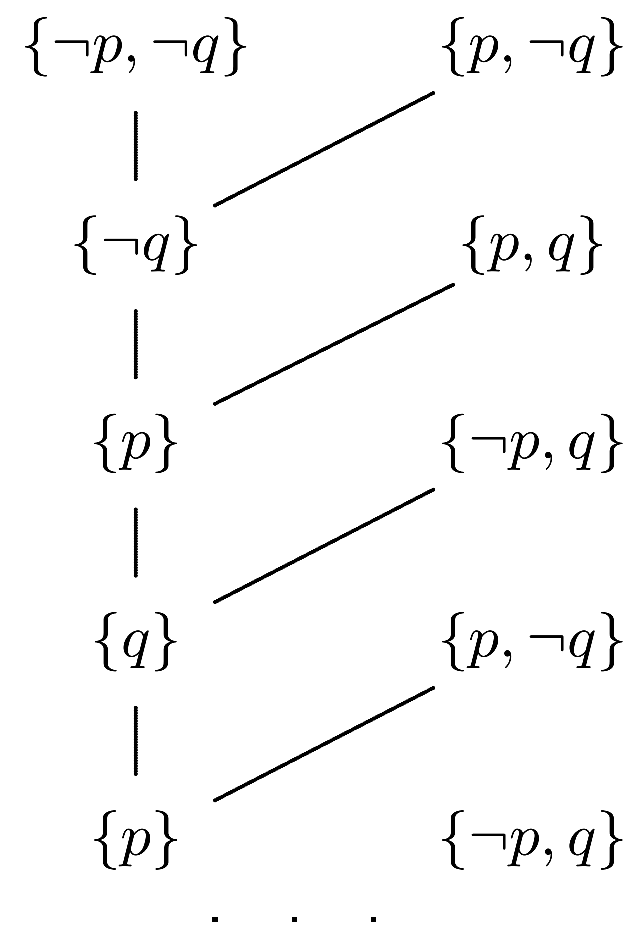
$$\{\{P(x,x)\},\{\neg P(x,y),\neg P(y,z),P(z,x)\},\{P(a,b)\},\{\neg P(b,a)\}\}$$



LI-resolution II

sound but not complete in general

Ex.: : $S = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$



LI-resolution is complete for Horn clauses

Horn clause

- max. one positive literal

  which of $\{\{p,q\},\{p,\neg q\},\{\neg p,q\},\{\neg p,\neg q\},\{p\}\}$ are Horn clauses?

- an alternative notation

  $\{p \leftarrow q\},\{p \rightarrow q\},\{true \rightarrow p\}$

- the Prolog notation

- rule `p :- q.`

  fact `p.`

  goal `?- p,q.`

LD-resolution

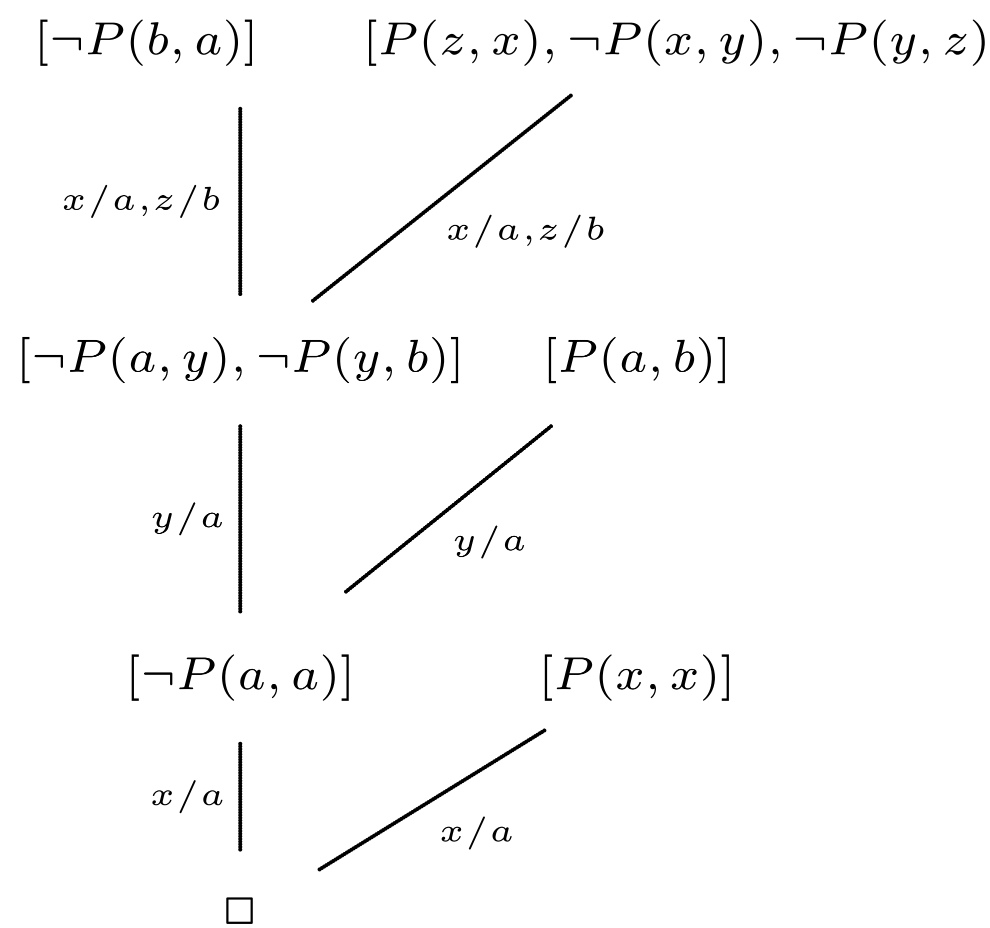
- from LI-resolution to an ordered resolution
- works with an *ordered clauses*; $[P(x), \neg R(x, f(y)), \neg Q(a)]$

If $G = [\neg A_1, \neg A_2, \dots, \neg A_n]$ and
$H = [B_0, \neg B_1, \neg B_2, \dots, \neg B_m]$ are ordered clauses and ϕ an mgu for B_0 and A_i),
then the *(ordered) resolvent* of G a H is the ordered clause
$[\neg A_1\phi, \neg A_2\phi, \dots, \neg A_{i-1}\phi, \neg B_1\phi, \neg B_2\phi, \dots, \neg B_m\phi, \neg A_{i+1}\phi, \dots, \neg A_n\phi]$

LD – Linear Definite

LD-resolution

$$\{[P(x,x)],[P(z,x),\neg P(x,y),\neg P(y,z)],[P(a,b)],[\neg P(b,a)]\}$$



LD-resolution is sound and complete for Horn clauses.

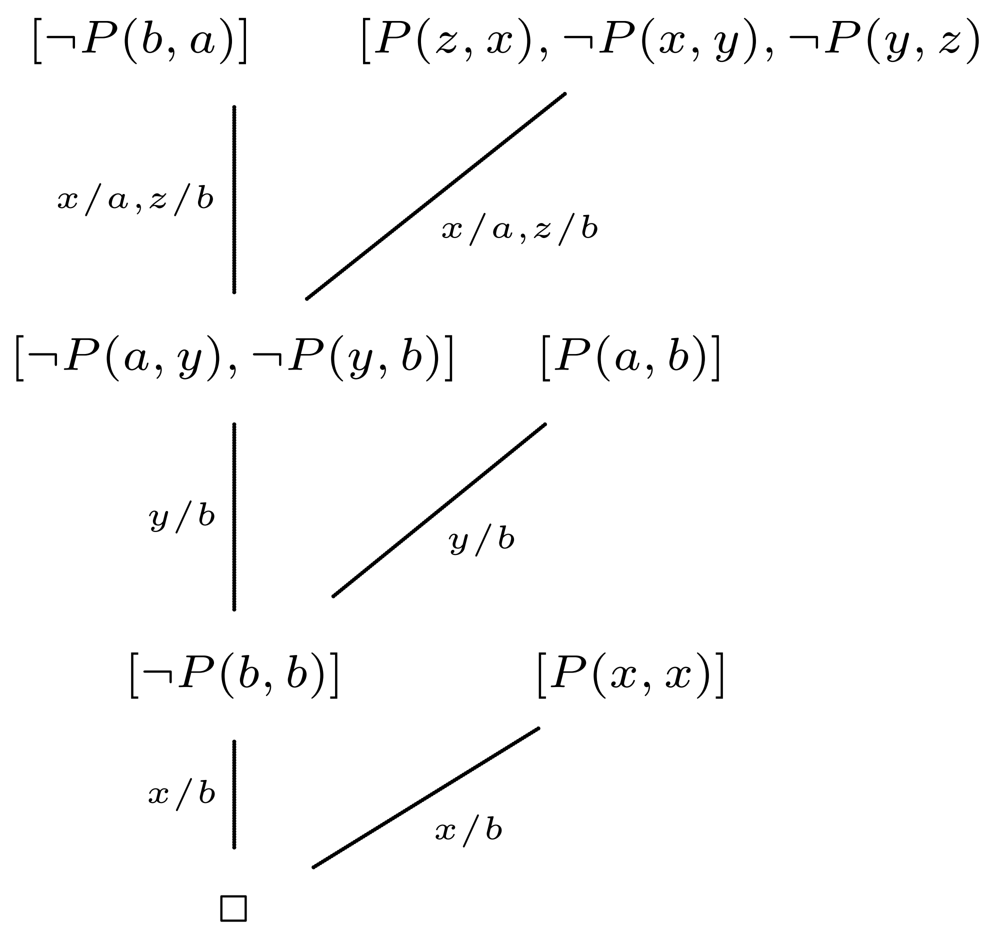
SLD-resolution

- LD–resolution with a selection rule
- A *selection rule* R s a function that chooses a literal from every nonempty ordered clause C.
- If no R is mentioned we assume that the standard one of choosing the leftmost literal is intended.
- Example: $G = [\neg A_1, \neg A_2, \ldots, \neg A_n]$,
  $H = [B_0, \neg B_1, \neg B_2, \ldots, \neg B_m]$,
  The resolvent of G and H for $\phi = mgu(B_0, A_1)$ is
  $[\neg B_1\phi, \neg B_2\phi, \ldots, \neg B_m\phi, \neg A_2\phi, \ldots, \neg A_n\phi]$

SLD-resolution is sound and complete for Horn clauses

SLD-resolution

selection rule = the leftmost literal



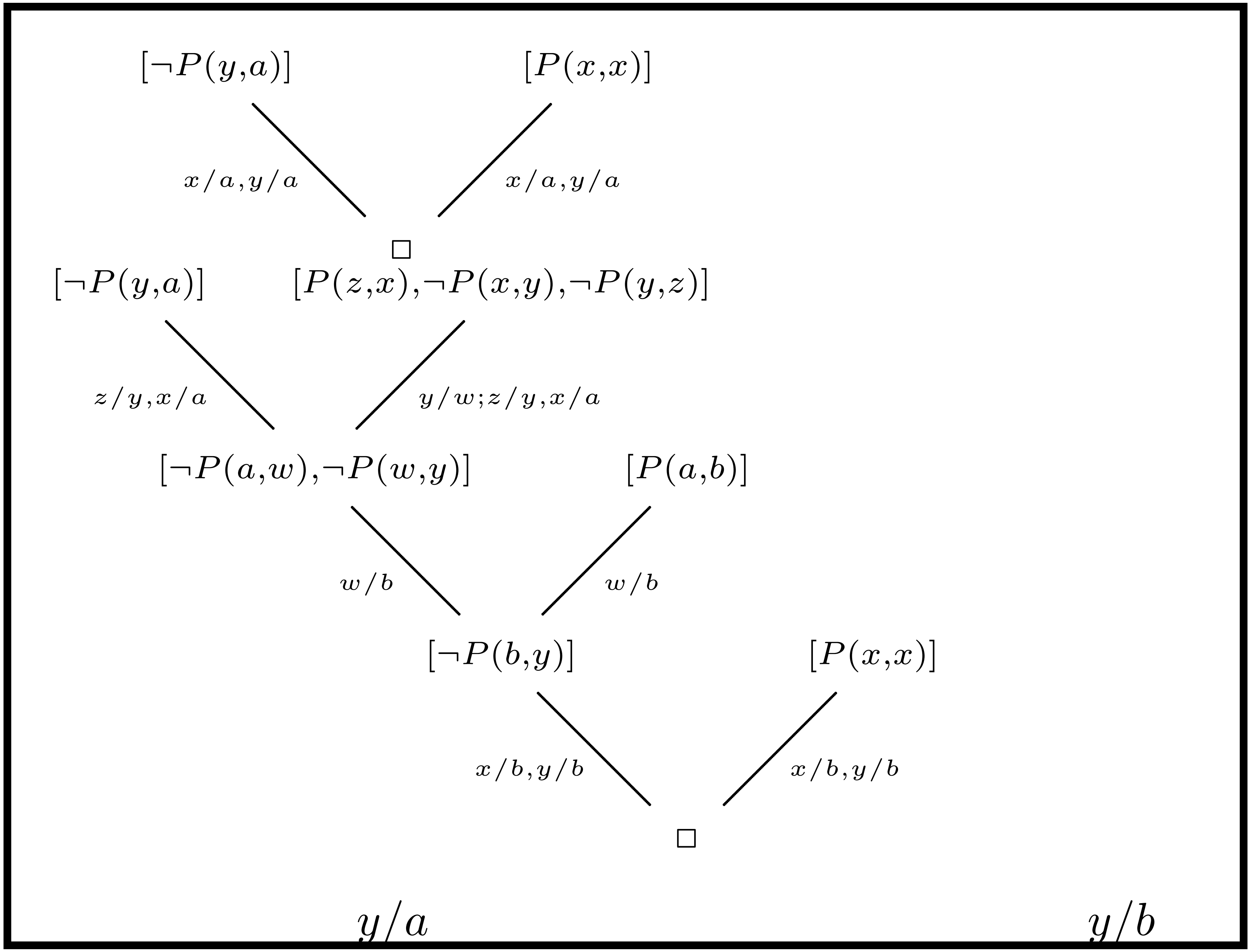
Example

For

$P = \{[P(a,b)], [P(x,x)], [P(z,x), \neg P(x,y), \neg P(y,z)]\}$,

find all solutions (i.e. substitutions of variables) of the goal

$[\neg P(y,a)]$

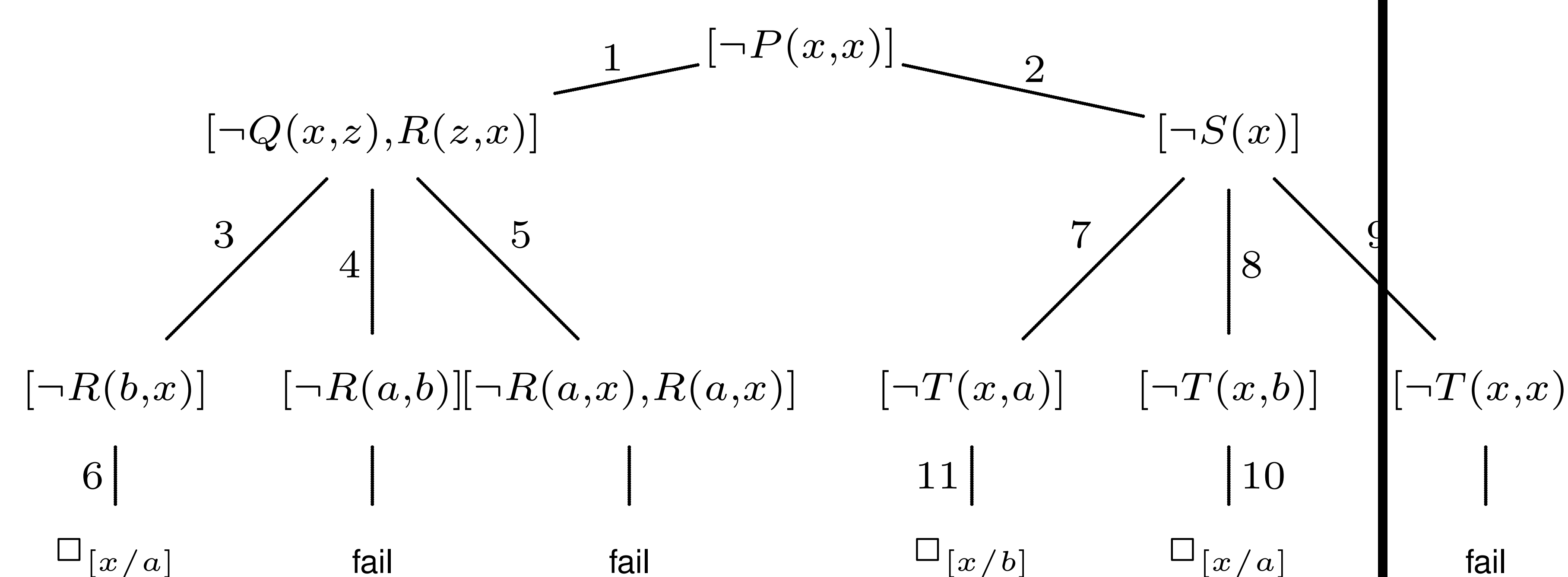

[¬P(y,a)]
[P(x,x)]
x/a,y/a
x/a,y/a
□
[¬P(y,a)]
[P(z,x),¬P(x,y),¬P(y,z)]
z/y,x/a
y/w;z/y,x/a
[¬P(a,w),¬P(w,y)]
[P(a,b)]
w/b
w/b
[¬P(b,y)]
[P(x,x)]
x/b,y/b
x/b,y/b
□
y/a
y/b

SLD-trees

all SLD-derivations for a given goal G and the program P

1. $[P(x,y),\neg Q(x,z),\neg R(z,y)]$
2. $[P(x,x),\neg S(x)]$
3. $[Q(x,b)]$
4. $[Q(b,a)]$
5. $[Q(x,a),\neg R(a,x)]$
6. $[R(b,a)]$
7. $[S(x),\neg T(x,a)]$
8. $[S(x),\neg T(x,b)]$
9. $[S(x),\neg T(x,x)]$
10. $[T(a,b)]$
11. $[T(b,a)]$

cíl: $[\neg P(x,x)]$



[¬P(x,x)]
1
2
[¬Q(x,z),R(z,x)]
[¬S(x)]
3
4
5
7
8
9
[¬R(b,x)]
[¬R(a,b)]
[¬R(a,x),R(a,x)]
[¬T(x,a)]
[¬T(x,b)]
[¬T(x,x)]
6
11
10
□[x/a]
fail
fail
□[x/b]
□[x/a]
fail