VI. Automatic Reasoning

1. Resolution

Task:

A_1: If the sun shines, Peter goes to the beach.

A_2: If Peter goes to the beach, he can swim.

A_3: Peter cannot swim at home.

Prove:

B: If the sun shines, then Peter does not stay at home.

Formalization:

A_1: $p \rightarrow q$

A_2: $q \rightarrow r$

A_3: $\neg(s \wedge r)$

B: $p \rightarrow \neg s$

–the sun shines : p

–Peter goes to the beach : q

–Peter can swim: r

–Peter stays at home: s

Preparing the proof

- Need: $p \to q,\quad q \to r,\quad \neg(s \wedge r) \Rightarrow\ p \to \neg s$
  - By definition, every interpretation (truth assignment) satisfying the antecedents also satisfies the consequence.
  - Or: none of interpretations (truth assignments) satisfying the antecedents can satisfy the negation of the consequence.
  - Or: $\{\ p \to q,\quad q \to r,\quad \neg(s \wedge r),\ \neg(p \to \neg s)\ \}$ is unsatisfiable
  - Or: $\{\ \neg p \vee q,\quad \neg q \vee r,\quad \neg s \vee \neg r,\quad p,\quad s\quad \}$ is unsatisfiable

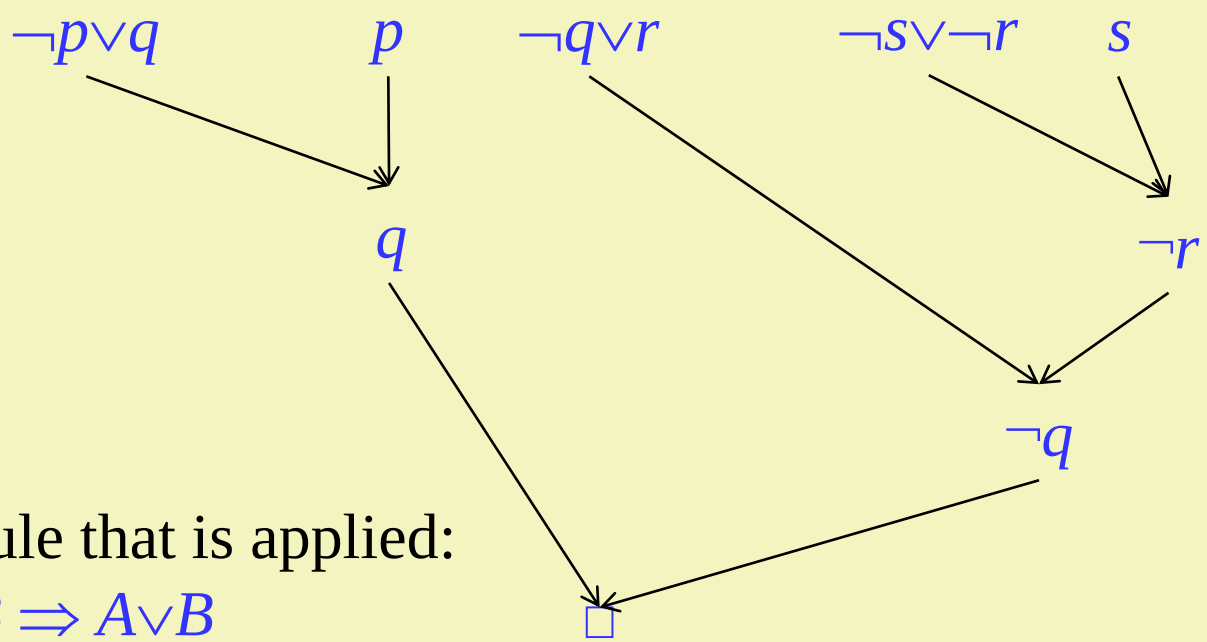
clause: literals connected by ‚or'
literal: variable or its negation

- We must to prove that in all interpretations (for all truth assignments) at least one of the clauses is *false*.

Indirect proof

- Assume that there exists an interpretation so that it satisfies all clauses, i.e. all clauses are *true*.
- For example the clause p and the clause $\neg p \vee q$ is also *true*.
- If p is *true*, $\neg p$ is *false*. In order to $\neg p \vee q$ be *true* the q must be also *true*.
- Thus the original clauses can be extended with the new clause q since it must be *true* like the original clauses in the given interpretation supposed in our assumption.

Resolution process

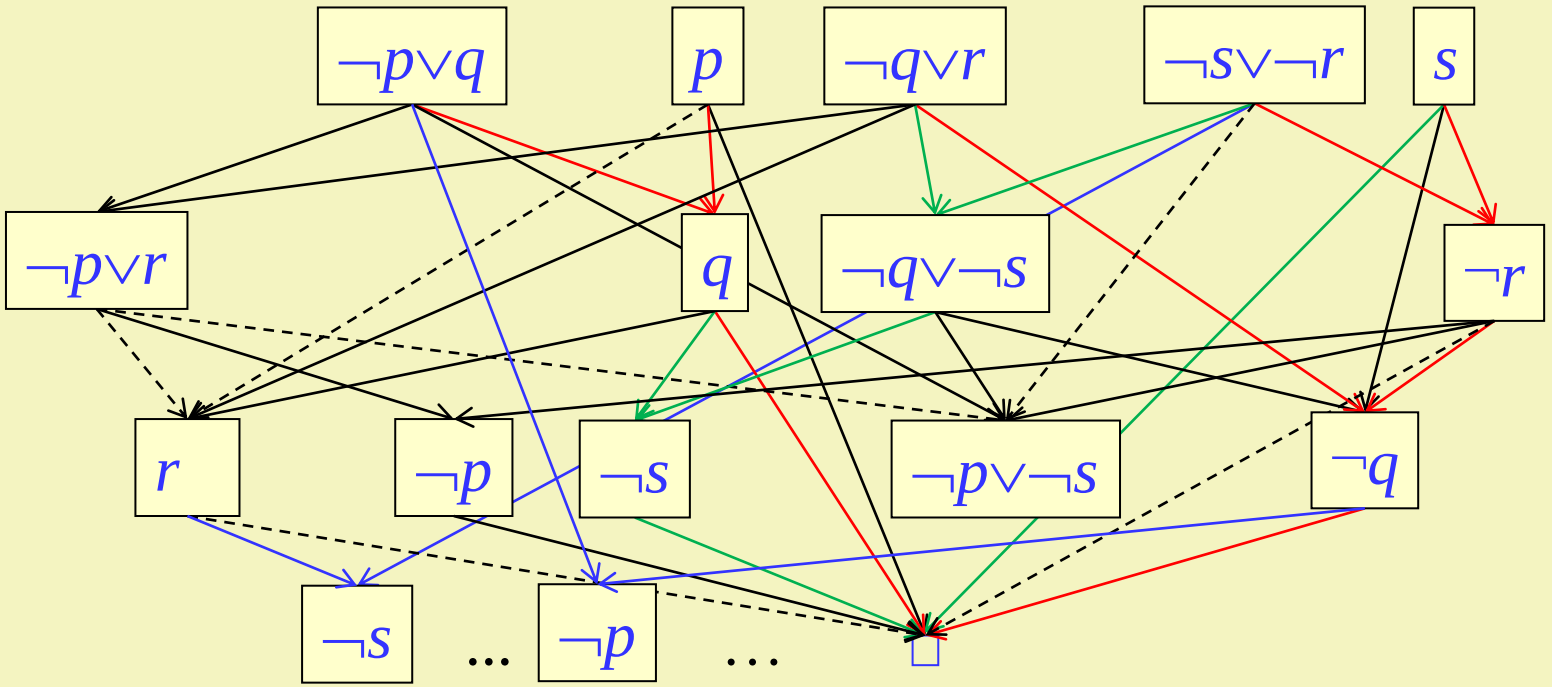


Inference rule that is applied:

$L \vee A, \neg L \vee B \Rightarrow A \vee B$

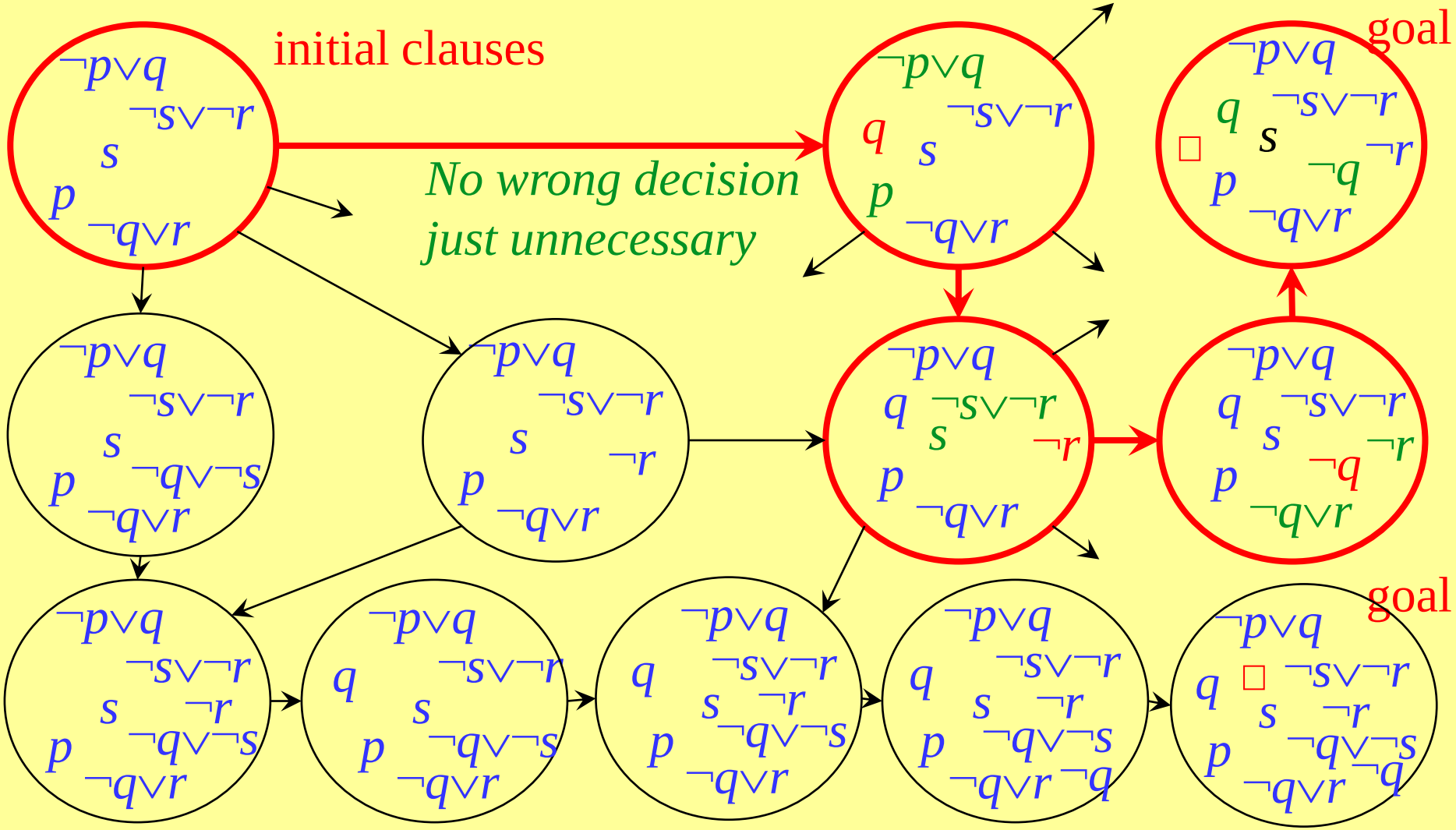
Thus if the sun shines, then Peter does not stay at home.

Refutation-, resolution graph



- Refutation graph: shows one deduction of the empty clause
- Resolution graph: shows the deductions of all possible clauses

Representation graph



DATA := *initial value*
while ¬ *termination condition*(DATA) **loop**
SELECT R FROM *rules that can be applied*
DATA := R(DATA)
endloop

Irrevocable search system

- global workspace: set of clauses
- initial value: set of original clauses derived from the task (axioms ⇒ target)
- termination condition:
  - success: empty clause
  - fail: no newer resolvent clause
- searching rule: creates a new resolvent
- control strategy: selects one of the searching rules
- heuristic: no idea

Example: Are physicians quacks?

A_1: Some patients trust all physicians.

A_2: Patients don't trust any quack.

Prove that

B : Physicians are not quacks.

Formalization:

$A_1 : \exists x\{P(x) \wedge \forall y[D(y) \rightarrow T(x,y)]\}$

$A_2 : \forall x\{P(x) \rightarrow \forall y[Q(y) \rightarrow \neg T(x,y)]\}$

$B : \forall x[D(x) \rightarrow \neg Q(x)]$

$P(x)$: x is a patient

$D(y)$: y is a physician

$Q(y)$: y is a quack

$T(x,y)$: x trusts y

Skolemized conjunctive normal form

A_1 : $\exists x\{P(x) \wedge \forall y[D(y) \rightarrow T(x,y)]\}$

$= \exists x\{P(x) \wedge \forall y[\neg D(y) \vee T(x,y)]\}$

$\approx P(\boldsymbol{a}) \wedge \forall y[\neg D(y) \vee T(\boldsymbol{a},y)]$

$P(a)\ ,\ \neg D(y) \vee T(a,y)$

logical law (equivalent transforming)

Skolemization (It holds the satisfiability.) *a* is a Skolem constant

A_2 : $\forall x\{\ P(x) \rightarrow \forall y[Q(y) \rightarrow \neg T(x,y)]\}$

$= \forall x\{\neg P(x) \vee \forall y[\neg Q(y) \vee \neg T(x,y)]\}$

$= \forall x\{\neg P(x) \vee \forall \boldsymbol{u}[\neg Q(\boldsymbol{u}) \vee \neg T(x,\boldsymbol{u})]\}$

$\neg P(x) \vee \neg Q(u) \vee \neg T(x,u)$

logical law

renaming distinguishes the variable from the variables of the other clauses

B: $\neg\forall x[\ D(x) \rightarrow \neg Q(x)]$

$=\neg\forall x[\neg D(x) \vee \neg Q(x)] = \exists x[D(x) \wedge Q(x)] \approx D(\boldsymbol{b}) \wedge Q(\boldsymbol{b})$

$D(b)\ ,\ Q(b)$

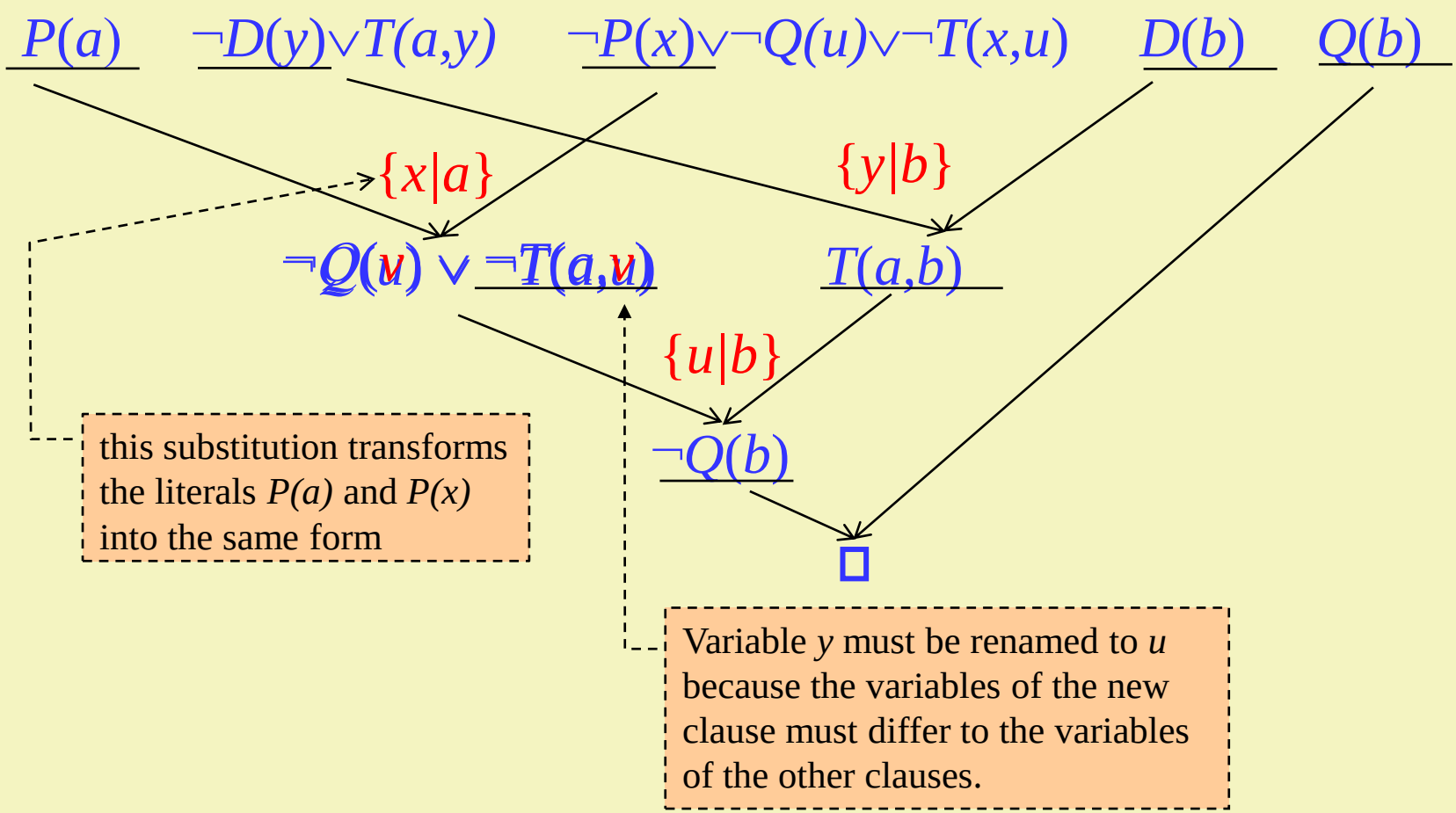
logical law

b is a Skolem constant

Conversion to set of clauses

1. Eliminate implication (→) and equivalence symbols (↔).
2. Reduce the scope of negation symbols.
3. Standardize (rename) variables so that each quantifier must have its own unique variable
4. Eliminate existential quantifiers.
   Instead of $\exists x P(x)$: $P(a)$ can be written where a is a Skolem-constant.
   Instead of $\forall x \exists y P(y)$: $\forall x P(g(x))$ where g is a Skolem-function.
5. Move all universal quantifiers to the front of the formula.
6. Put formula in conjunctive normal form.
7. Rename the variables so that no variable symbol appears in more than one clause.
8. Form the set of clauses.

Resolution process



Resolvable clauses and their resolvent

- The clauses C_1 and C_2 are resolvable
  - if they contain a complementary literal pair (C_1 contains some positive instances of a predicate symbol, C_2 contains some negative instances of the same predicate symbol),
  - so that few of its positive instances of C_1 and few of its negative instances of C_2 can be unified.

$C_1 = \quad P(t_{11},...,t_{1n}) \vee ... \vee P(t_{r1},...,t_{rn}) \vee C_1'$

$C_2 = \neg P(u_{11},...,u_{1n}) \vee ... \vee \neg P(u_{s1},...,u_{sn}) \vee C_2'$

C_1' and C_2' may be empty but they can contain $P()$ or $\neg P()$.

- Let δ be the most general unifier of the instances $P(t_{11},...,t_{1n}), ... , P(t_{r1},...,t_{rn}), P(u_{11},...,u_{1n}), ... , P(u_{s1},...,u_{sn})$.
  The resolvent of C_1 and C_2 : $R(C_1 , C_2) = C_1'\delta \vee C_2'\delta$.
- The empty clause ($\square$) is an unsatisfiable clause by definition.

DATA := *initial value*
while ¬ *termination condition*(DATA) **loop**
    SELECT R FROM *rules that can be applied*
    DATA := R(DATA)
endloop

Resolution

In order to prove $A_1, A_2, ..., A_n \Rightarrow B$ it is enough to show that the clause form of formulas $A_1, A_2, ..., A_n, \neg B$ are unsatisfiable.

1. *CLAUSES* := clauses of $A_1, A_2, ..., A_n, \neg B$
2. **loop**
3. **if** $\square \in$ *CLAUSES* **then return** *unsatisfiable*
4. **if** there are no resolvable $C_1, C_2 \in$ *CLAUSES* so that $R(C_1, C_2)$ is unknown (not included in *CLAUSES*) **then return** *satisfiable*
5. **select** $C_1, C_2 \in$ *CLAUSES* where $R(C_1, C_2)$ is unknown
6. *CLAUSES* := *CLAUSES* $\cup$ $R(C_1, C_2)$
7. **endloop**

Example: There are no professors

Problem: *Professors teach all people who cannot learn alone.*
There is no professor who teaches a man who can learn alone.
Thus there are no professors.

Formalization:

$T(x,y)$ ~ x teaches y
$P(x)$ ~ x is a professor

$\forall x\,[P(x) \rightarrow \forall y(\neg T(y,y) \rightarrow T(x,y))]$
$\neg\exists x\,[P(x) \wedge \exists y(\,T(y,y) \wedge T(x,y))]$
$\neg\exists x\,P(x)$

$\neg P(x_1) \vee \underline{T(y_1,y_1) \vee T(x_1,y_1)}$ $\quad \neg P(x_2) \vee \underline{\neg T(y_2,y_2) \vee \neg T(x_2,y_2)}$ $\quad P(a)$

$\{x_1 | y_2,\ x_2 | y_2,\ y_1 | y_2\}$

$\neg P(y_2)$

$\{y_2 | a\}$

$\square$

Features of resolution

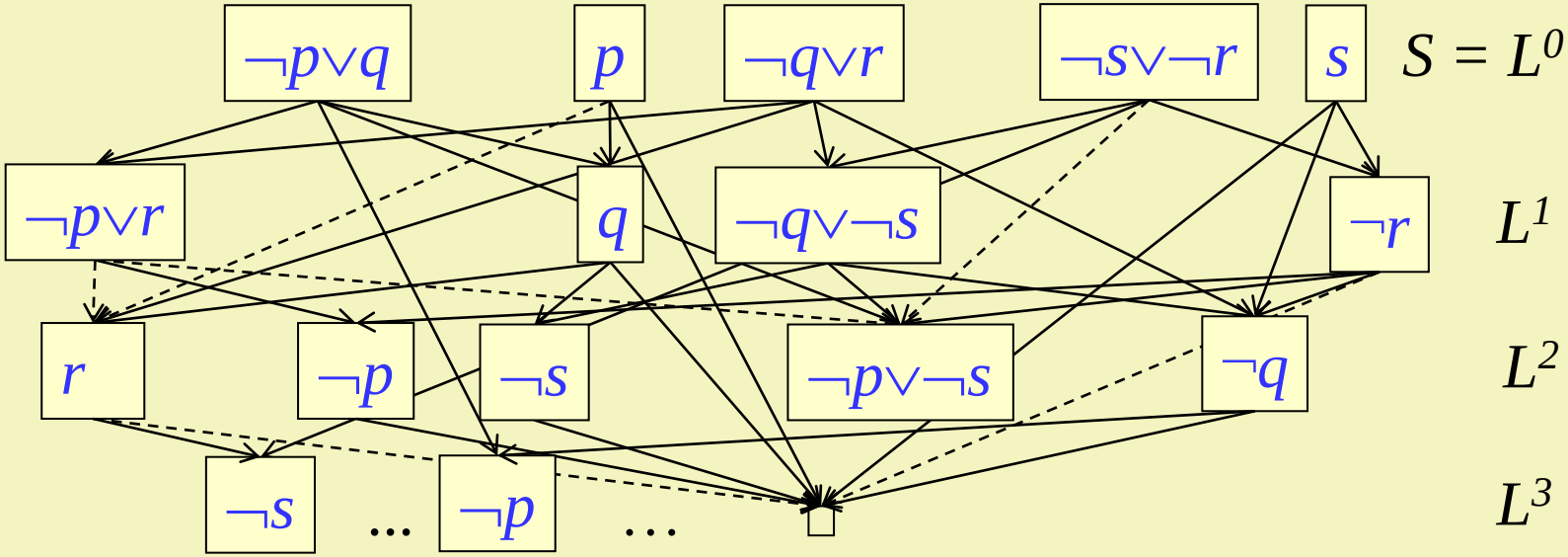
- The resolution refutation is sound: if it finds the empty clause, then the original clauses must be unsatisfiable.
- The resolution refutation is complete: the empty clause can be derived from an unsatisfiable set of clauses.
  - But in first order logic it is not guaranteed that the refutation terminates, so the unsatisfiablity problem is only a partial decidable: $\{\neg P(x), P(y)\vee\neg P(f(y)), P(a)\}$
- The resolution is nondeterministic. In one step, there may be
  - several resolvable clause pairs
  - several complementary predicate symbols in the selected clause pair
  - several occurrences of the selected predicate symbol
    $\{P(x,f(a))\vee P(x,f(y))\vee Q(y), \neg P(z,f(a))\vee\neg Q(z), P(u,f(a))\vee\neg Q(a)\}$

Resolution strategies

- The resolution strategies can
  - restrict the set of derivable resolvents
  - give an order of the construction of resolvents.
- These strategies are secondary control strategies because they can be applied to only a clause based representation.
- The completeness of the resolution may be lost under a restricting strategy:
  - The empty clause cannot be deduced always.

Ordering strategies

- In the *breadth-first strategy*, the resolvents are produced level by level according to the resolution graph.
  - The deepest parent of an i^{th} level resolvent is on $(i-1)^{th}$ level.
  - Since each level contains only finite clauses the resolution must find the empty clause in finite steps if the empty clause is derivable.



Ordering strategies

- The *length of clauses strategy* (length=number of literals), prefers the resolvent that has got the shortest parents.
  - The pair of clauses C_1,C_2 is shorter than the pair of clauses D_1, D_2 if the shortest clause of the pair C_1,C_2 is shorter than the shortest clause of the pair D_1, D_2. If these lengths are identical, then the lengths of the other two clauses must be compared.

Restricting strategies

- The resolution graph can be restricted:
  - Unit-preference strategy COMPLETE with only Horn clauses
    one parent is always a unit clause (including one literal)
  - Linear-input strategy COMPLETE with only Horn clauses
    one parent belongs to the base set, the other parent is the previous resolvent (except in the first step)
  - Ancestry-filtered form strategy COMPLETE
    one parent either is in the base set or that is an ancestor of the other parent, the other parent is the previous resolvent
  - Set-of-support strategy COMPLETE if $S-T$ satisfiable
    one parent belongs to the clauses derived from the given subset T of the original clauses S

Simplification strategies

- Elimination of tautologies
  A clause that is always true (ex. $P(x) \vee \neg P(x)$) is useless in the derivation of the empty clause (that is always false).

- Elimination by subsumption
  The clause C subsumes the clause D if there exits a substitution α so that $C\alpha$ is a part of D (ex. $P(x)$ subsumes $P(a) \vee Q(z)$). In this case the clause D can be removed.

- Elimination of clauses with extraneous literal
  A literal is extraneous if its predicate symbol does not occur in other clauses with opposite sign.

Procedural attachment

- Procedures can be attached to certain symbols in order to compute their values with respect to a given model.
  For example:
- A ground instance of a function symbol can be substituted with the constant symbol that represents the value of the very instance in the given model.
  - Example: If the function symbol *sub* is the subtraction on integers, then *sub*(*4*,*1*) can be written as *3*.
- The truth of a ground instance of a predicate symbol can be evaluated based on its meaning.
  - Example: If in the formula *EQ*(*3*,*2*) the predicate symbol *EQ* is the equality, then *EQ*(*3*,*2*) is *false*.

Extracting answers from resolution refutations

"If Fido goes wherever John goes and John is at school, where is Fido?"

Formalization:

$AT(y,x)$ ~ y is at place x

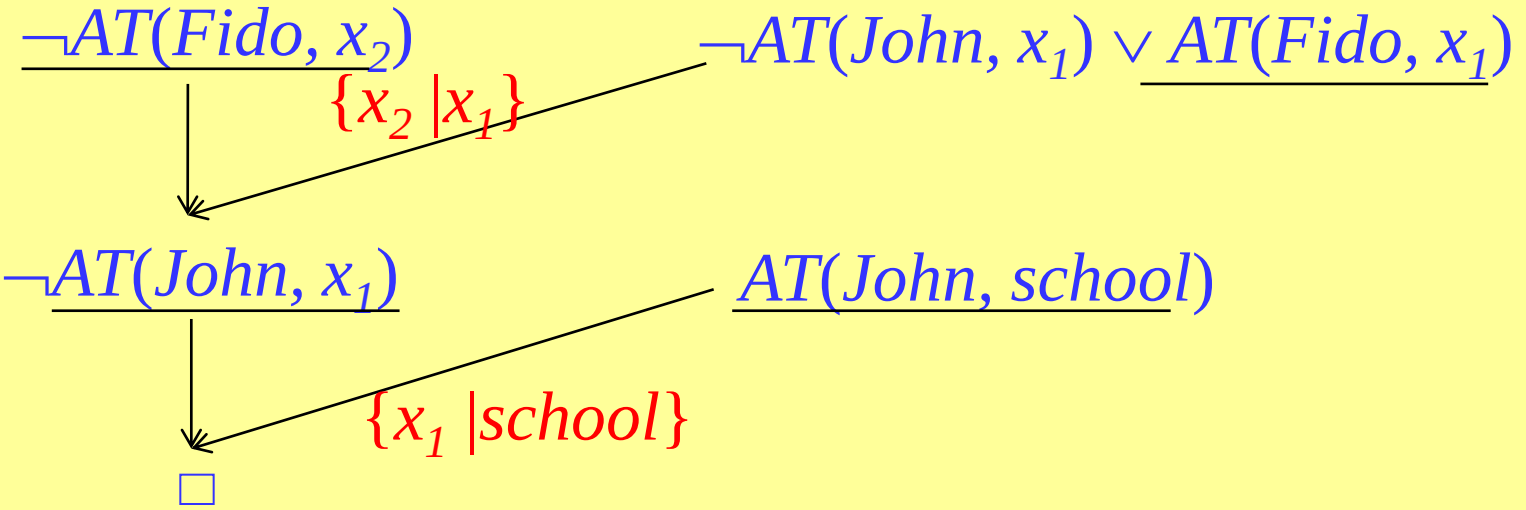
Are there any place for Fido?

$\forall x[AT(John,x) \rightarrow AT(Fido,x)]$

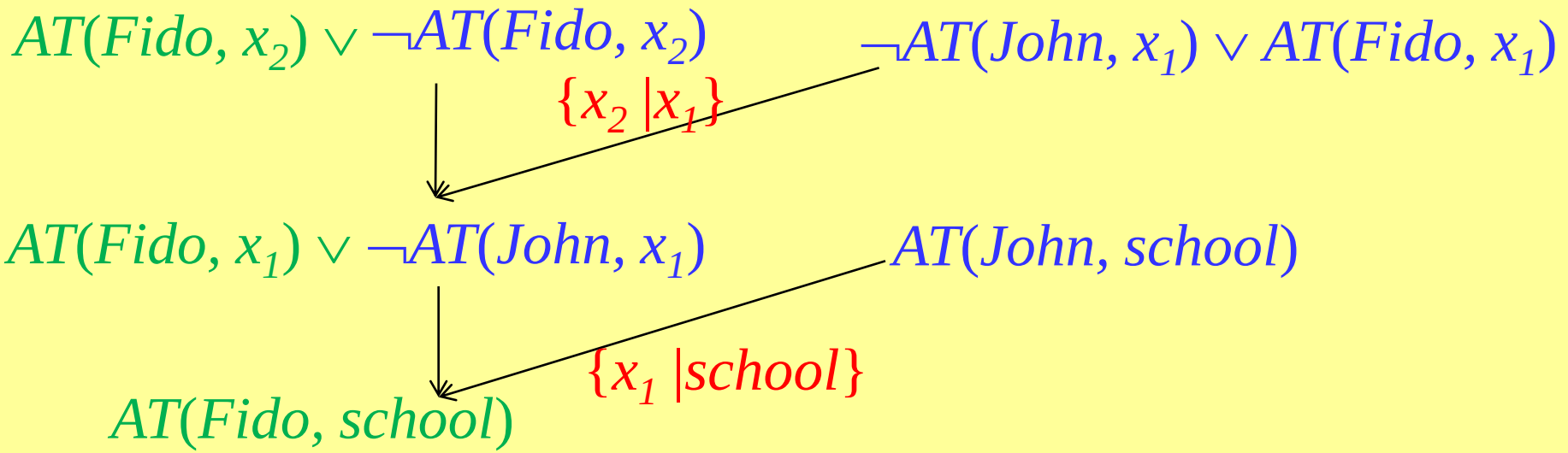
$AT(John, school)$

$\Rightarrow \quad \exists x AT(Fido,x)$

Refutation graph



Answering graph



The answer extraction process

1. The original question (who, which, where, when, how much) must be substituted with a goal statement „there exists the answer”.
2. The refutation graph must be found.
3. The clauses resulting from the negation of the goal formula are converted into tautologies by appending to them their own negations.
4. Following the structure of the refutation graph, perform the same resolutions as before until some clause is obtained at the root.
5. Use the clause at the root as an answer statement.

A logical representation of Hanoi tower problem

- Fact
  - $H(1,i,j,k,t(i,j).nil)$
- Rule
  - $H(n-1,i,k,j,y) \wedge H(1,i,j,k,t(i,j).nil) \wedge H(n-1,k,j,i,z) \rightarrow H(n,i,j,k, conc(y,t(i,j).nil,z))$
- Goal
  - $\exists x\, H(2,3,1,2,x)$

$t(i,j)$ function symbol denotes the move of a disc from the peg i to the peg j

nil is the empty sequence

'.' is a two-arguments function symbol in infix form that can create a sequence from one move and a sequence.

$conc(x, y, z)$ function symbol denotes the concatenation of the sequences x, y és z

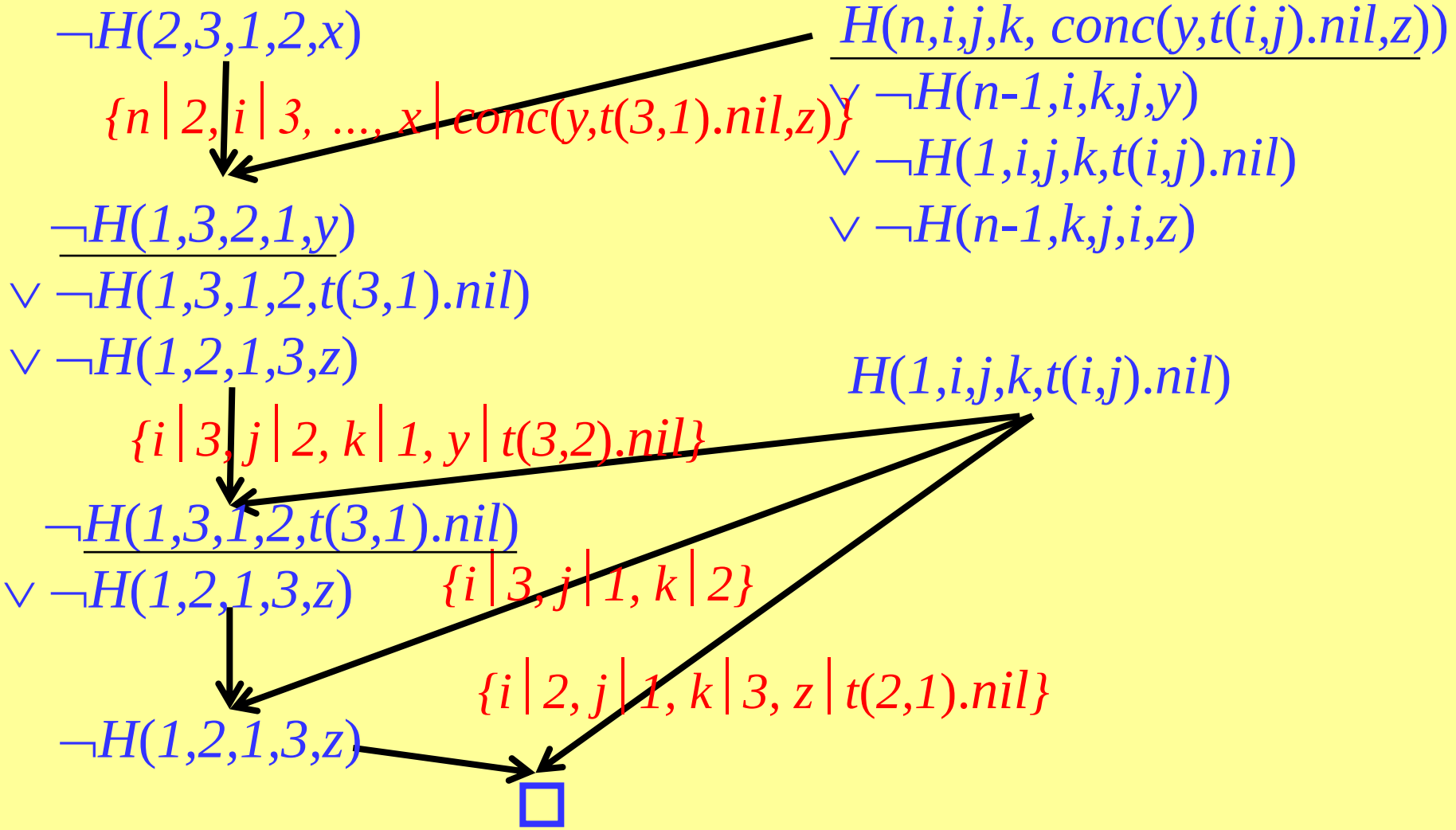
$H(n,i,j,k,x)$ predicate symbol is true if the moves of the sequence x can put n pieces discs from the peg i to the peg j

Variables are bound by universal quantifiers, except the goal

Clauses of Hanoi tower problem

- $H(1,i,j,k,t(i,j).nil)$
- $\neg H(n-1,i,k,j,y) \vee \neg H(1,i,j,k,t(i,j).nil) \vee \neg H(n-1,k,j,i,z) \vee H(n,i,j,k, conc(y,t(i,j).nil,z))$
- $\neg\, H(2,3,1,2,x)$

Refutation graph



Answering graph