Knowledge representation and reasoning (KRR)

- First-order logic: 句法 syntax and 语义 semantics
- Resolution-based inference procedure

What is KRR?

Symbolic encoding of propositions (命题) believed by some agent and their manipulation (操作) to produce representations of propositions that are believed by the agent but not explicitly represented

An example

- Explicitly represented beliefs:
  $GradStu(Ann)$, $GradStu(Bob)$,
  $\forall x($ GradStu(x) $\rightarrow$ Student(x) $)$
- Implicitly represented beliefs:
  $Student(Ann)$, $Student(Bob)$,
  $\forall x(\neg$ Student(x) $\rightarrow \neg$ GradStu(x)$)$

We need knowledge to answer questions

Could a crocodile run a steeplechase? [Levesque 88]

- Yes
- No

The intended thinking: short legs, tall hedges $\Rightarrow$ No!

Yet another example

Consider a question about materials:

> The large ball crashed right through the table because it was made of XYZZY. What was made of XYZZY?
>
> - the large ball
> - the table

Now suppose that you learn some facts about XYZZY.

1. It is a trademarked (商标) product of the Dow Chemical Company.
2. It is usually white, but there are green and blue varieties.
3. It is ninety-eight percent air, making it lightweight and buoyant (轻飘).
4. It was first discovered by a Swedish inventor, Carl Georg Munters.

Ask: At what point does the answer stop being just a guess?

Why KRR?

- KR hypothesis: any artificial intelligent system is knowledge-based 假设 任何人工智能系统都是基于知识的
- Knowledge-based system: system with structures that
  - can be interpreted propositionally and 可用命题解释
  - determine the system behavior

  such structures are called its knowledge base (KB)
- Knowledge-based system most suitable for open-ended tasks 开放式
- Hallmark of knowledge-based system: cognitive penetrability, *i.e.*, actions depend on beliefs, including implicitly represented beliefs 认知渗透

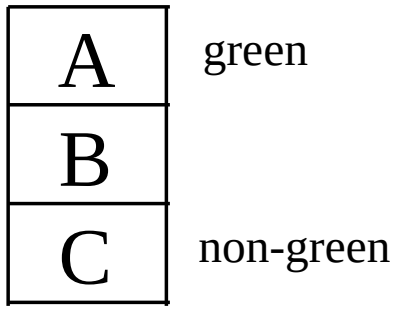
KRR and logic

Logic is the main tool for KRR, because logic studies

- How to formally represent agent's beliefs
- Given the explicitly represented beliefs, what are the implicitly represented beliefs

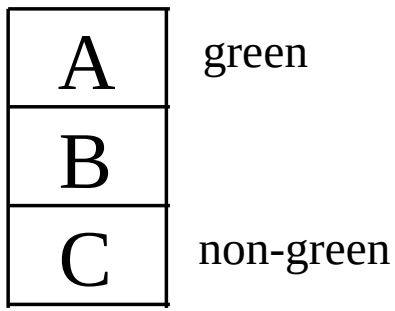
There are many kinds of logics. In this course, we will use first-order logic (FOL) as the tool for KRR

A blocks world example



- Given the scene, human can easily draw the conclusion "there is a green block directly on top of a non-green block"
- How can a machine do the same?

Formalization in FOL



- $S = \{On(a, b), On(b, c), Green(a), \neg Green(c)\}$
- $\alpha = \exists x \exists y[Green(x) \wedge \neg Green(y) \wedge On(x, y)]$
- S logically entails α

An example

- Tony, Mike, and John belong to the Alpine Club.
- Every member of the Alpine Club who is not a skier is a mountain climber.
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
- Tony likes rain and snow.
- Is there a member of the Alpine Club who is a mountain climber but not a skier?

An example (cont'd)

- Intelligence is needed to answer the question
- Can we make machines answer the question?
- A possible approach
  - First, translate the sentences and question into FOL formulas
    - Of course, this is hard, and we do not have a good way to automate this step
  - Second, check if the formula of the question is logically entailed by the formulas of the sentences
    - We will show that there are ways to automate this step

自动化

Alphabet

- Individuals (constants or 0-ary functions):
  - tony, mike, john
  - rain, snow
- Types (unary predicates):
  - $A(x)$ means that x belongs to Alpine Club
  - $S(x)$ means that x is a skier
  - $C(x)$ means that x is a mountain climber
- Relationships (binary predicates):
  - $L(x, y)$ means that x likes y

Basic facts

- Tony, Mike, and John belong to the Alpine Club.
  $A(tony), A(mike), A(john)$
- Tony likes rain and snow.
  $L(tony, rain), L(tony, snow)$

Complex facts

- Every member of the Alpine Club who is not a skier is a mountain climber.
  $\forall x(A(x) \wedge \neg S(x)) \rightarrow C(x)$
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
  $\forall x(C(x) \rightarrow \neg L(x, rain))$
  $\forall x(\neg L(x, snow) \rightarrow \neg S(x))$
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
  $\forall x(L(tony, x) \rightarrow \neg L(mike, x))$
  $\forall x(\neg L(tony, x) \rightarrow L(mike, x))$
- Is there a member of the Alpine Club who is a mountain climber but not a skier?
  $\exists x(A(x) \wedge C(x) \wedge \neg S(x))$

Alphabet

Logical symbols (fixed meaning and use):

- Punctuation: (,),,,,.
- Connectives and quantifiers: $=, \neg, \wedge, \vee, \forall, \exists$

连接词 量词 '='表示两个符号代表同一物体

- Variables: $x, x_1, x_2, \ldots, x', x'', \ldots, y, \ldots, z, \ldots$

Non-logical symbols (domain-dependent meaning and use):

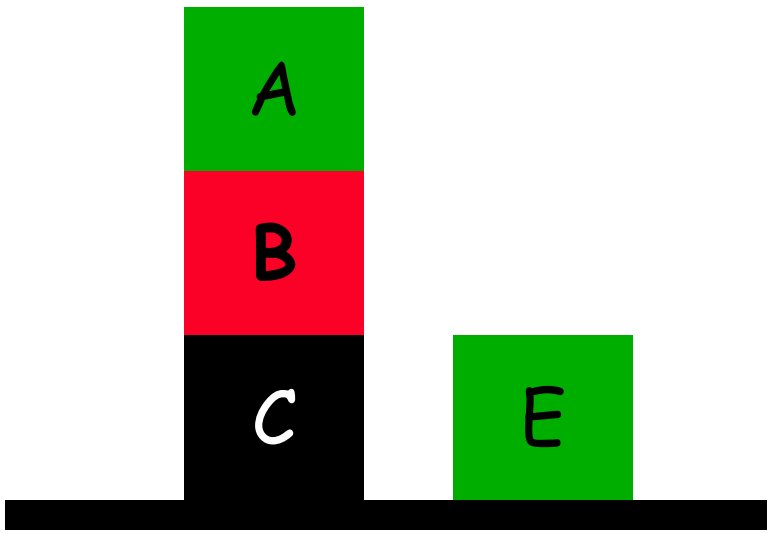
- Predicate symbols
  - arity: number of arguments
  - arity 0 predicates: propositional symbols

谓词 命题 0元谓词是命题符号

- Function symbols
  - arity 0 functions: constant symbols

A blocks world example

Environment



Language (Syntax)

- Constants: a,b,c,e
- Functions:
  - No function
- Predicates:
  - on: binary
  - above: binary
  - clear: unary
  - ontable: unary

Terms

- Every variable is a term (项)
- If $t_1, \ldots, t_n$ are terms and f is a function symbol of arity n, then $f(t_1, \ldots, t_n)$ is a term

Formulas

- If $t_1, \ldots, t_n$ are terms and P is a predicate symbol of arity n, then $P(t_1, \ldots, t_n)$ is an atomic formula
- If t_1 and t_2 are terms, then $(t_1 = t_2)$ is an atomic formula
- If α and β are formulas, and v is a variable, then $\neg\alpha, (\alpha \wedge \beta), (\alpha \vee \beta), \exists v.\alpha, \forall v.\alpha$ are formulas

Notation符号

- Occasionally add or omit (,) 忽略
- Use [,] and $\{,\}$
- Abbreviation: $(\alpha \rightarrow \beta)$ for $(\neg\alpha \vee \beta)$ 缩写
- Abbreviation: $(\alpha \leftrightarrow \beta)$ for $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- Predicates: mixed case capitalized, *e.g.*, Person, OlderThan
- Functions (and constants): mixed case uncapitalized, *e.g.*, john, father,

Variable scope

- Free and bound occurrences of variables

自由出现的x　　约束出现的x（前有存在符号）

- *e.g.*, $P(x) \wedge \exists x[P(x) \vee Q(x)]$

句子是没有自由变量的公式

- A sentence: formula with no free variables

代换　　将自由出现的v由项t代替

- Substitution: $\alpha[v/t]$ means α with all free occurrences of the v replaced by term t

- In general, $\alpha[v_1/t_1, \ldots, v_n/t_n]$

Interpretations

解释

An interpretation is a pair $\Im = \langle D, I \rangle$

- D is the domain, can be any non-empty set
- I is a mapping from the set of predicate and function symbols
- If P is a predicate symbol of arity n, $I(P)$ is an n-ary relation over D, *i.e.*, $I(P) \subseteq D^n$ P是n元谓词，I(P)是D上的n元关系
  - If p is a 0-ary predicate symbol, *i.e.*, a propositional symbol, $I(p) \in \{true, false\}$
- If f is a function symbol of arity n, $I(f)$ is an n-ary function over D, *i.e.*, $I(f) : D^n \rightarrow D$
  - If c is a 0-ary function symbol, *i.e.*, a constant symbol, $I(c) \in D$

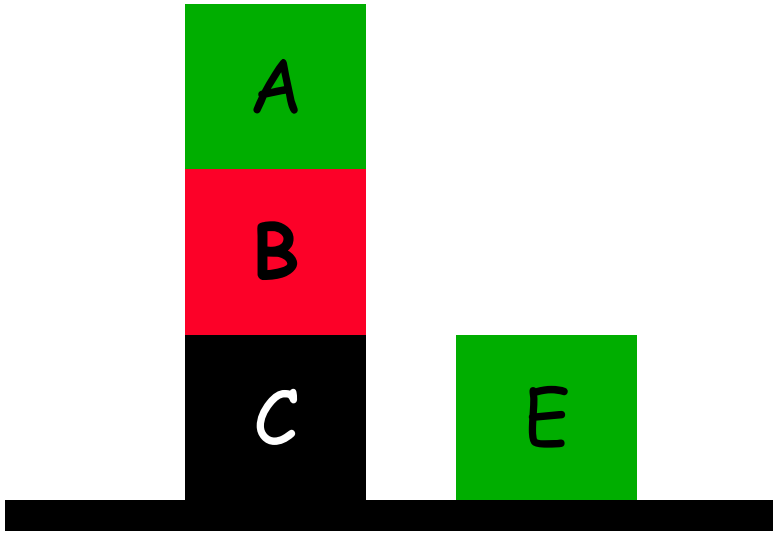
Blocks world example

- D = {A, B, C, E}
- Φ(a) = A, Φ(b) = B, Φ(c) = C, Φ(e) = E.
- Ψ(on) = {(A,B),(B,C)}

on是2元谓词，I(on)是D上的2元关系

- Ψ(above) = {(A,B),(B,C),(A,C)}
- Ψ(clear)={A,E}
- Ψ(ontable)={C,E}

Environment



Denotation of terms

- Terms denote (表示) elements of the domain
- A variable assignment μ is a mapping from the set of variables to the domain D
- $\|v\|_{\Im,\mu} = \mu(v)$
- $\|f(t_1, \ldots, t_n)\|_{\Im,\mu} = I(f)(\|t_1\|_{\Im,\mu}, \ldots, \|t_n\|_{\Im,\mu})$

Satisfaction: atomic formulas

可満足符号

$\mathfrak{S}, \mu \models \alpha$ is read "$\mathfrak{S}, \mu$ satisfies α"

- $\mathfrak{S}, \mu \models P(t_1, \ldots, t_n)$ iff $\langle \|t_1\|_{\mathfrak{S},\mu}, \ldots, \|t_n\|_{\mathfrak{S},\mu} \rangle \in I(P)$
- $\mathfrak{S}, \mu \models (t_1 = t_2)$ iff $\|t_1\|_{\mathfrak{S},\mu} = \|t_2\|_{\mathfrak{S},\mu}$

Satisfaction: propositional connectives

- $\Im, \mu \models \neg\alpha$ iff $\Im, \mu \not\models \alpha$
- $\Im, \mu \models (\alpha \wedge \beta)$ iff $\Im, \mu \models \alpha$ and $\Im, \mu \models \beta$
- $\Im, \mu \models (\alpha \vee \beta)$ iff $\Im, \mu \models \alpha$ or $\Im, \mu \models \beta$

Satisfaction: quantifiers 量词

d赋值给v

$\mu\{d; v\}$ denotes a variable assignment just like μ, except that it maps v to d

- $\Im, \mu \models \exists v.\alpha$ iff for some $d \in D$, $\Im, \mu\{d; v\} \models \alpha$
- $\Im, \mu \models \forall v.\alpha$ iff for all $d \in D$, $\Im, \mu\{d; v\} \models \alpha$

句子是没有自由变量的公式

Let α be a sentence. Then whether $\Im, \mu \models \alpha$ is independent of μ. Thus we simply write $\Im \models \alpha$

Blocks world example

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$
- $\Phi(a) = \underline{A}$, $\Phi(b) = \underline{B}$, $\Phi(c) = \underline{C}$, $\Phi(e) = \underline{E}$.
- $\Psi(on) = \{(\underline{A},\underline{B}),(\underline{B},\underline{C})\}$
- $\Psi(above) = \{(\underline{A},\underline{B}),(\underline{B},\underline{C}),(\underline{A},\underline{C})\}$
- $\Psi(clear)=\{\underline{A},\underline{E}\}$
- $\Psi(ontable)=\{\underline{C},\underline{E}\}$

$\forall X,Y.\ on(X,Y) \rightarrow above(X,Y)$

✓ $X=\underline{A}$, $Y=\underline{B}$

✓ $X=\underline{C}$, $Y=\underline{A}$

✓ ...

$\forall X,Y.\ above(X,Y) \rightarrow on(X,Y)$

✓ $X=\underline{A}$, $Y=\underline{B}$

× $X=\underline{A}$, $Y=\underline{C}$

Blocks world example

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$
- $\Phi(a) = \underline{A}$, $\Phi(b) = \underline{B}$, $\Phi(c) = \underline{C}$, $\Phi(e) = \underline{E}$.
- $\Psi(on) = \{(\underline{A},\underline{B}),(\underline{B},\underline{C})\}$
- $\Psi(above) = \{(\underline{A},\underline{B}),(\underline{B},\underline{C}),(\underline{A},\underline{C})\}$
- $\Psi(clear)=\{\underline{A},\underline{E}\}$
- $\Psi(ontable)=\{\underline{C},\underline{E}\}$

$\forall X \exists Y.\ (clear(X) \vee on(Y,X))$

- ✓ $X=\underline{A}$
- ✓ $X=\underline{C}$, $Y=\underline{B}$
- ✓ ...

$\exists Y \forall X.(clear(X) \vee on(Y,X))$

- × $Y=\underline{A}$? No! ($X=\underline{C}$)
- × $Y=\underline{C}$? No! ($X=\underline{B}$)
- × $Y=\underline{E}$? No! ($X=\underline{B}$)
- × $Y=\underline{B}$? No! ($X=\underline{B}$)

Satisfiability 可满足

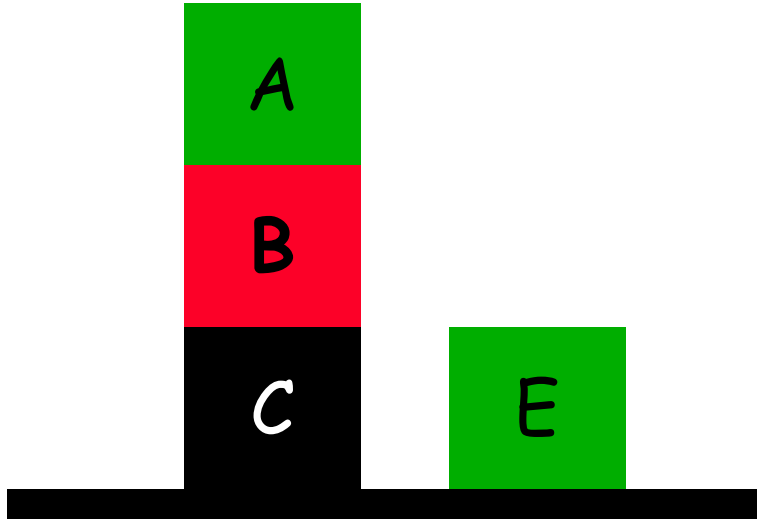
- Let S be a set of sentences
- $\Im \models S$, read $\Im$ satisfies S, if for every $\alpha \in \Im$, $\Im \models \alpha$
- If $\Im \models S$, we say $\Im$ is a model of S
- We say that S is satisfiable if there is $\Im$ s.t. $\Im \models S$, and
- *e.g.*, is $\{\forall x(P(x) \rightarrow Q(x)), P(a), \neg Q(a)\}$ satisfiable?

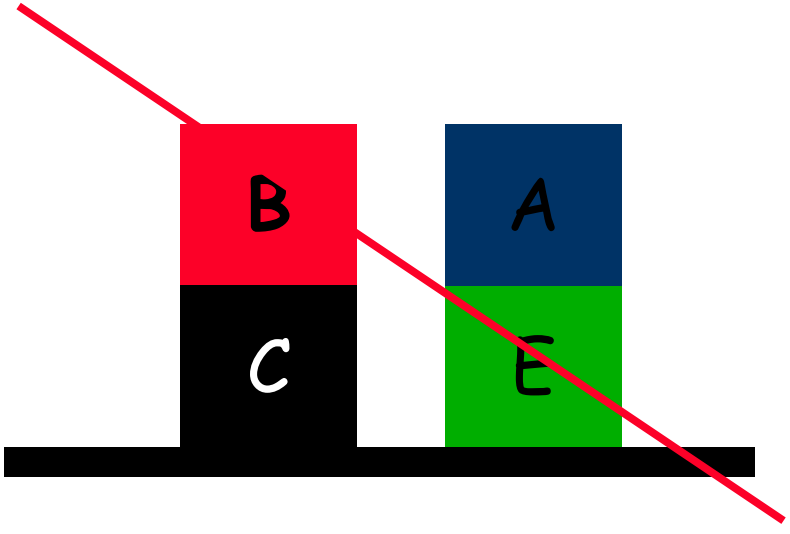
  不可满足，因为P(a)→Q(a)，而给定的是~Q(a)，所以不可满足

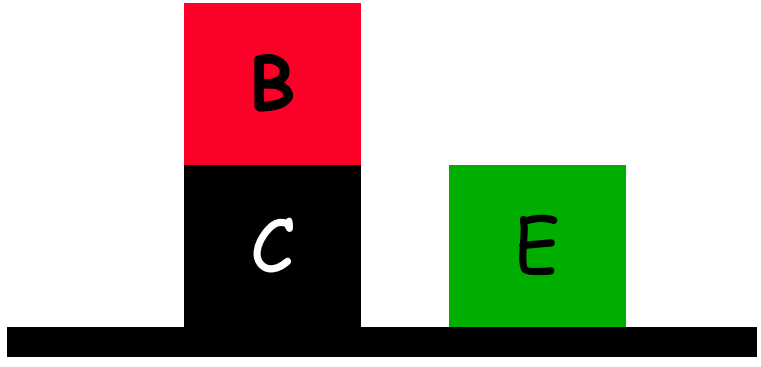
Blocks world example

KB

1. on(b,c)
2. clear(e)







Logical entailment

- $S \models \alpha$ iff for every $\Im$, if $\Im \models S$ then $\Im \models \alpha$
- $S \models \alpha$ is read: S entails α or α is a logical consequence of S

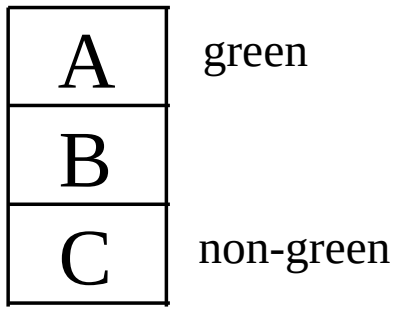
任何解释满足空集，也就是说任何解释满足a，所以称a是有效的，a为真

- A special case: $\emptyset \models \alpha$, simply written $\models \alpha$, read "α is valid"
- Note that $\{\alpha_1, \ldots, \alpha_n\} \models \alpha$ iff $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha$ is valid iff $\alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\alpha$ is unsatisfiable

KB |= a 等价于 KB^~a 不可满足

- Alpine Club example
  - Let KB be the set of sentences, and α be the question
  - We want to know if $KB \models \alpha$?

Blocks world example cont'd



- $S = \{On(a,b), On(b,c), Green(a), \neg Green(c)\}$
- $\alpha = \exists x \exists y [Green(x) \wedge \neg Green(y) \wedge On(x,y)]$

对任意解释满足S，该解释也满足a，那么S |= a

- We prove that $S \models \alpha$

Logical entailment: examples

- $\forall x A \vee \forall x B \models \forall x(A \vee B)$ 错
- Does $\forall x(A \vee B) \models \forall x A \vee \forall x B$ 对
- $\exists x(A \wedge B) \models \exists x A \wedge \exists x B$
- Does $\exists x A \wedge \exists x B \models \exists x(A \wedge B)$?
- $\exists y \forall x A \models \forall x \exists y A$
- Does $\forall x \exists y A \models \exists y \forall x A$? 错

The only way to prove that $KB \not\models \alpha$ is to give an interpretation satisfying KB but not α.

Alpine Club example cont'd

- Suppose that we had been told that Mike likes whatever Tony dislikes, but we had not been told that Mike dislikes whatever Tony likes.
- Can we still claim that there is a member of the Alpine Club who is a mountain climber but not a skier?
- No. We give an interpretation which satisfies the modified KB but not f as follows: Let $D = \{T, M, J, R, S\}$. Let $I(tony) = T, I(mike) = M, I(john) = J, I(rain) = R, I(snow) = S$. Let $I(A) = \{T, M, J\}, I(S) = \{T, M, J\}, I(C) = \emptyset$, $I(L) = \{(T, R), (T, S), (T, T), (M, M), (M, S), (M, J), (J, S)\}$.

Logical entailment and knowledge-based systems

- Start with KB representing explicit beliefs, usually what the agent has been told or has learned
- Implicit beliefs: $\{\alpha \mid KB \models \alpha\}$
- Actions depend on implicit beliefs, rather than explicit beliefs

Inference procedure

- We want a mechanical procedure to check if $KB \models \alpha$
- Called an inference 推理 procedure
- Sound if whenever it says yes, then $KB \models \alpha$
- Complete if whenever $KB \models \alpha$, then it says yes

Resolution-based Inference procedure

- Resolution is a rule of inference
- Resolution-based inference procedure: refutation 驳斥
- We begin with the propositional case
- Then proceed to the first-order case

Clausal form

- A literal is an atomic formula or its negation, *e.g.*, $p, \neg p$
- A clause is a disjunction (析取（或）) of literals, written as the set of literals
  - *e.g.*, $p \vee \neg r \vee s$, written $(p, \neg r, s)$
- A special case: empty clause $()$ (空的子句（）表示假), representing false
- A formula is a conjunction (合取（且）) of clauses, written as the set of clauses

Resolution rule of inference

- From the two clauses $\{p\} \cup c_1$ and $\{\neg p\} \cup c_2$,
  infer the clause $c_1 \cup c_2$
  推断
- $c_1 \cup c_2$ is called the resolvent of input clauses wrt the atom p
  归结
- *e.g.*, (p) and $(\neg p)$ resolve to $()$,
  p和~p归结得到空集，即假
  (w, r, q) and $(w, s, \neg r)$ resolve to (w, q, s) wrt r

- **Proposition.** $\{p\} \cup c_1, \{\neg p\} \cup c_2 \models c_1 \cup c_2$
  Proof:

Derivation 推导

A derivation of a clause c from a set S of clauses is a sequence $c_1, c_2, \ldots, c_n$ of clauses, where $c_n = c$, and for each c_i, either

- $c_i \in S$, or
- c_i is a resolvent of two earlier clauses in the derivation

We write $S \vdash c$ if there is a derivation of c from S

Soundness of derivations

可靠性

- **Theorem.** If $S \vdash c$, then $S \models c$
  Proof:
  - Let $c_1, c_2, \ldots, c_n$ be a derivation of c from S
  - We prove by induction on i that for all $1 \leq i \leq n$, $S \models c_i$.
    归纳法
- However, the converse does not hold in general
  e.g., $(p) \models (p, q)$, but $(p) \not\vdash (p, q)$ 逆命题不成立

Soundness and completeness of refutations

Theorem. $S \vdash ()$ iff $S \models ()$ iff S is unsatisfiable

We will not prove the completeness part

Resolution-based inference procedure: refutation

$KB \models \alpha$ iff $KB \wedge \neg\alpha$ is unsatisfiable

Thus to check if $KB \models \alpha$,

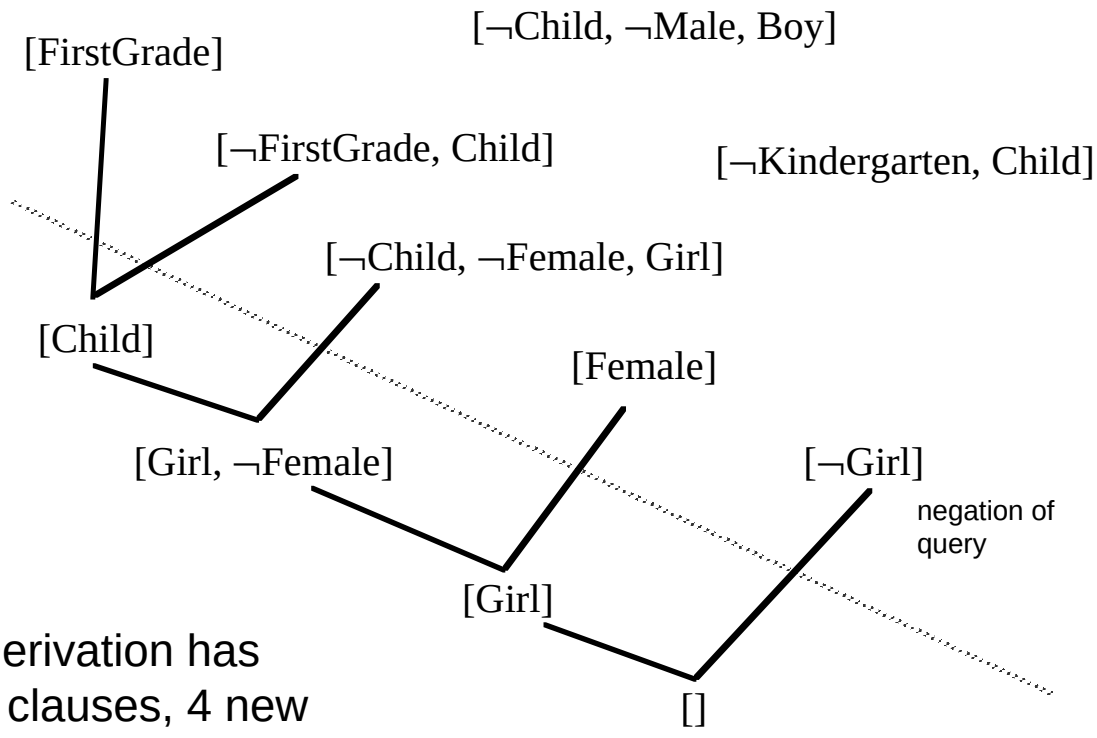
- put KB and $\neg\alpha$ into clausal form to get S,
  子句
- check if $S \vdash ()$

Refutation example 1

KB

FirstGrade
FirstGrade $\supset$ Child
Child $\wedge$ Male $\supset$ Boy
Kindergarten $\supset$ Child
Child $\wedge$ Female $\supset$ Girl
Female

Show that KB |= Girl



Derivation has
9 clauses, 4 new

Refutation example 2

KB |= Mail 等价于 KB^~Mail 不可满足

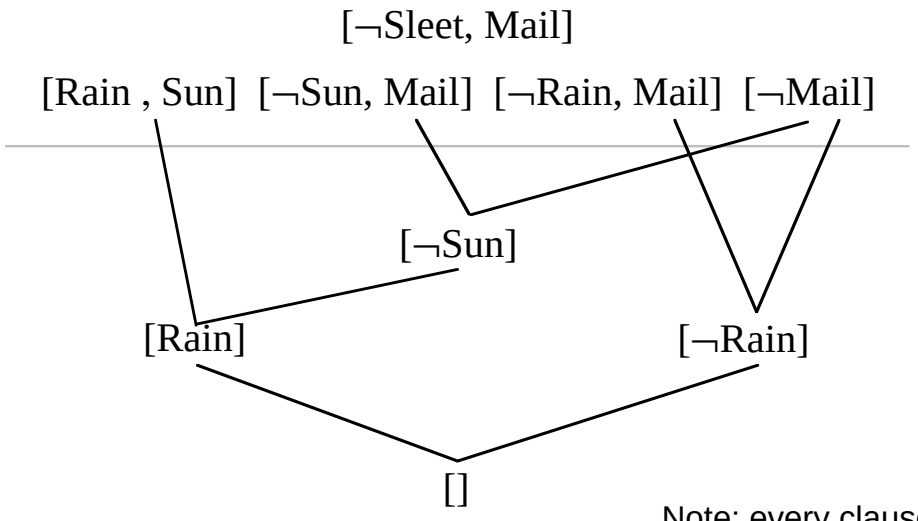
KB

$(\text{Rain} \vee \text{Sun})$

$(\text{Sun} \supset \text{Mail})$

$((\text{Rain} \vee \text{Sleet}) \supset \text{Mail})$

Show KB |= Mail



Note: every clause not in S has 2 parents

Similarly KB |≠ Rain

枚举

Can enumerate all resolvents given ¬Rain, and [] will not be generated

The first-order case

We need

- A way of converting KB and f (the query) into clausal form
- A way of doing resolution even when we have variables. This needs unification
  统一

Conversion to Clausal Form

1. Eliminate Implications.
2. Move Negations inwards (and simplify $\neg\neg$).
3. Standardize Variables.
4. Skolemize.
5. Convert to Prenex Form.
6. Distribute disjunctions over conjunctions.
7. Flatten nested conjunctions and disjunctions.
8. Convert to Clauses.

Skolemization Skolem范式

Consider $\exists y.Elephant(y) \wedge Friendly(y)$

- This asserts that there is some individual that is both an elephant and friendly.
- To remove the existential (存在量词), we invent a name for this individual, say a. This is a new constant symbol not equal to any previous (以前) constant symbols:
  $Elephant(a) \wedge Friendly(a)$
- This is saying the same thing, since we do not know anything about the new constant a.
- It is essential that the introduced symbol a is new. Else we might say more than the existential formula.

Skolemization

Now consider $\forall x \exists y. Loves(x, y)$.

- This formula claims that for every x there is some y that x loves (perhaps a different y for each x).
- Replacing the existential by a new constant won't work: $\forall x. Loves(x, a)$, because this asserts that there is a particular individual a loved by every x.

为正确地转化被全称量词限制的存在量词，使用函数而不是常量

- To properly convert existential quantifiers scoped by universal quantifiers we must use functions not just constants.
- In this case x scopes y, so we must replace y by a function of x: $\forall x. Loves(x, g(x))$, where g is a new function symbol.

使用函数项g(x)替代存在变量y

- This formula asserts that for every x there is some individual (given by $g(x)$) that x loves. $g(x)$ can be different for each x.

Skolemization examples

- $\forall x, y, z \exists w. R(x, y, z, w) \Longrightarrow \forall x, y, z. R(x, y, z, h_1(x, y, z))$
- $\forall x, y \exists w. R(x, y, g(w)) \Longrightarrow \forall x, y. R(x, y, g(h_2(x, y)))$
- $\forall x, y \exists w \forall z. R(x, y, w) \wedge Q(z, w) \Longrightarrow$
  $\forall x, y, z. R(x, y, h_3(x, y)) \wedge Q(z, h_3(x, y))$

A conversion example

$$\forall x\{P(x) \rightarrow [\forall y(P(y) \rightarrow P(f(x,y))) \wedge \neg\forall y(\neg Q(x,y) \wedge P(y))]\}$$

1. Eliminate implications using $A \rightarrow B \Leftrightarrow \neg A \vee B$

蕴涵

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

Move negations inwards

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

否定词内移

2. Move negations inwards using
   - $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$, $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
   - $\neg\exists x.A \Leftrightarrow \forall x.\neg A$, $\neg\forall x.A \Leftrightarrow \exists x.\neg A$, $\neg\neg A \Leftrightarrow A$

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists y(Q(x,y) \vee \neg P(y))]\}$$

Standardize Variables

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists y(Q(x,y) \vee \neg P(y))]\}$$

3. Standardize Variables (Rename variables so that each quantified variable is unique)

标准化变量（重命名变量使得每个变量的名称都不相同）

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists z(Q(x,z) \vee \neg P(z))]\}$$

Skolemize

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists z(Q(x,z) \vee \neg P(z))]\}$$

4. Skolemize (Remove existential quantifiers by introducing new function symbols)

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge (Q(x, g(x)) \vee \neg P(g(x)))]\}$$

Convert to prenex form

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \vee \neg P(g(x)))]\}$$

5. Convert to prenex form. (Bring all quantifiers to the front – only universals, each with different name)

全称量词前移

$$\forall x\forall y\{\neg P(x) \vee [(\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \vee \neg P(g(x)))]\}$$

Disjunctions over conjunctions

$$\forall x \forall y \{\neg P(x) \vee [(\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \vee \neg P(g(x)))]\}$$

析取分配到合取（最后我们要得到子句的合取）

6. Disjunctions over conjunctions using

$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$

$$\forall x \forall y \{(\neg P(x) \vee \neg P(y) \vee P(f(x,y))) \wedge (\neg P(x) \vee Q(x,g(x)) \vee \neg P(g(x)))\}$$

7.把多余的小括号去掉

Convert to Clauses

$$\forall x \forall y \{(\neg P(x) \vee \neg P(y) \vee P(f(x,y))) \wedge (\neg P(x) \vee Q(x, g(x)) \vee \neg P(g(x)))\}$$

8. Convert to Clauses (remove quantifiers and break apart conjunctions).

转化为子句

a) $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$

b) $\neg P(x) \vee Q(x, g(x)) \vee \neg P(g(x))$

Unification

- Can the clauses $(P(john), Q(fred), R(x))$ and $(\neg P(y), R(susan), R(y))$ be resolved?
  子句的归结可通过代入来实现
- Once reduced to clausal form, all remaining variables are universally quantified.
- So, implicitly the clause $(P(john), Q(fred), R(x))$ represents $(P(john), Q(fred), R(john))$, $(P(john), Q(fred), R(fred))$, ...
- So there is a specialization of $(P(john), Q(fred), R(x))$ that can be resolved with a specialization of $(\neg P(y), R(susan), R(y))$
- In particular, $(P(john), Q(fred), R(john))$ can be resolved with $(\neg P(john), R(susan), R(john))$, producing $(Q(fred), R(john), R(susan))$

Unification

我们希望代入的数据具有一般性，而不是特殊的

- We want to be able to match conflicting literals, even when they have variables.
- This matching process automatically determines whether or not there is a specialization that matches.
- But, we don't want to over specialize!

Unification

- Consider $(\neg P(x), S(x), Q(fred))$ and $(P(y), R(y))$
- We need to unify $P(x)$ and $P(y)$. How do we do this?
- Possible resolvants:
  - $(S(john), Q(fred), R(john))\{x = john, y = john\}$
  - $(S(sally), Q(fred), R(sally))\{x = sally, y = sally\}$
  - $(S(x), Q(fred), R(x))\{y = x\}$
- The last resolvant is most-general, the other two are specializations. We want the most general clause for use in future resolution steps.

Substitution 代换

- Unification is a mechanism for finding a most general matching
- A key component of unification is substitution.
  使用项t实例化变量V，但项t不能含有变量V
- A substitution is a finite set of equations of the form $V = t$ where V is a variable and t is a term not containing V. (t might contain other variables).
- We can apply a substitution $\sigma = \{V_1 = t_1, \ldots, V_n = t_n\}$ to a formula f to obtain a new formula $f\sigma$ by simultaneously同时 replacing every variable V_i by term t_i.
- *e.g.*, $P(x, g(y, z))\{x = y, y = f(a)\} \Longrightarrow P(y, g(f(a), z))$
- Note that the substitutions are not applied sequentially, *i.e.*, the first y is not subsequently replaced by $f(a)$. 同时代换

Composition of substitutions

- We can compose two substitutions θ and σ to obtain a new substitution $\theta\sigma$
- Let $\theta = \{x_1 = s_1, x_2 = s_2, \ldots, x_m = s_m\}$, $\sigma = \{y_1 = t_1, y_2 = t_2, \ldots, y_k = t_k\}$
- Step 1. Get $S = \{x_1 = s_1\sigma, x_2 = s_2\sigma, \ldots, x_m = s_m\sigma, y_1 = t_1, y_2 = t_2, \ldots, y_k = t_k\}$
- Step 2. Delete any identities, *i.e.*, equations of the form $V = V$.
- Step 3. Delete any equation $y_i = s_i$ where y_i is equal to one of the x_j in θ. Why?

Composition example

- Let $\theta = \{x = f(y), y = z\}$, $\sigma = \{x = a, y = b, z = y\}$
- Step 1. Get $S = \{x = f(b), y = y, x = a, y = b, z = y\}$
- Step 2. Delete $y = y$. 删除V=V形式的
- Step 3. Delete $x = a$. x已经被f(b)代入，所以删除x = a
- The result is $S = \{x = f(b), y = b, z = y\}$

Note on substitutions

- The empty substitution $\epsilon = \{\}$ is also a substitution, and we have $\theta\epsilon = \theta$.
- More importantly, substitutions when applied to formulas are associative: $(f\theta)\sigma = f(\theta\sigma)$
  关联的
- Composition is simply a way of converting the sequential application of a series of substitutions to a single substitution.

Unifiers 合一项

对于公式f和g，如果存在一个代入p，使得fp = gp，则说f和g在语法上相似

- A unifier of two formulas f and g is a substitution σ that makes f and g syntactically identical.
- Note that not all formulas can be unified – substitutions only affect variables.
- *e.g.*, $P(f(x), a)$ and $P(y, f(w))$ cannot be unified, as there is no way of making $a = f(w)$ with a substitution.

不能用一个项代入另一个项

MGU

A substitution σ of two formulas f and g is a Most General Unifier (MGU) if

最一般的合一项

- σ is a unifier.
- For every other unifier θ of f and g there must exist a third substitution λ such that $\theta = \sigma\lambda$.

This says that every other unifier is "more specialized" than σ.

The MGU of a pair of formulas f and g is unique up to renaming.

在重命名前是唯一的

MGU example

- $P(f(x), z)$ and $P(y, a)$
- $\sigma = \{y = f(a), x = a, z = a\}$ is a unifier, but not an MGU
- $\theta = \{y = f(x), z = a\}$ is an MGU
- $\sigma = \theta\lambda$, where $\lambda = \{x = a\}$

一个代入p不是MGU，因为存在代入a和代入b，将b代入a可得到代入p

Computing MGUs

- The MGU is the "least specialized" way of making atomic formulas with variables match.
- We can compute MGUs.
- Intuitively we line up the two formulas and find the first sub-expression where they disagree.
- The pair of subexpressions where they first disagree is called the disagreement set.
- The algorithm works by successively fixing disagreement sets until the two formulas become syntactically identical.

Computing MGUs

Given two atomic formulas f and g

1. $k = 0$; $\sigma_0 = \{\}$; $S_0 = \{f, g\}$
2. If S_k contains an identical pair of formulas, stop and return σ_k as the MGU of f and g.
3. Else find the disagreement set $D_k = \{e_1, e_2\}$ of S_k
4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let $\sigma_{k+1} = \sigma_k\{V = t\}$; $S_{k+1} = S_k\{V = t\}$; $k = k + 1$; Goto 2
5. Else stop, f and g cannot be unified.

Computing MGU examples

第一个不一致有{f(a), y}，代入{y = f(a)}，S = {P(f(a), g(x)), P(f(a), f(a))}
第二个不一致有{g(x), f(a)}，不存在相应的代入，所以不存在MGU

1. $P(f(a), g(x))$ and $P(y, y)$
2. $P(a, x, h(g(z)))$ and $P(z, h(y), h(y))$
3. $P(x, x)$ and $P(y, f(y))$

第一个不一致有{z, a}，代入{z = a}，S = {P(a, x, h(g(a)), P(a, h(y), h(y))}
第二个不一致有{x, h(y)}，代入{z = a, x = h(y)}，S = {P(a, h(y), h(g(a)), P(a, h(y), h(y))}
第三个不一致有{y, g(a)}, 代入{z = a, x = h(g(a)), y = g(a)}，所以MGU为{z = a, x = h(g(a)), y = g(a)}
S = {P(a, h(g(a)), h(g(a)), P(a, h(g(a)), h(g(a)))}

First-order Resolution

From the two clauses $\{\rho_1\} \cup c_1$ and $\{\neg\rho_2\} \cup c_2$, where there exists a MGU σ for ρ_1 and ρ_2, infer the clause $(c_1 \cup c_2)\sigma$

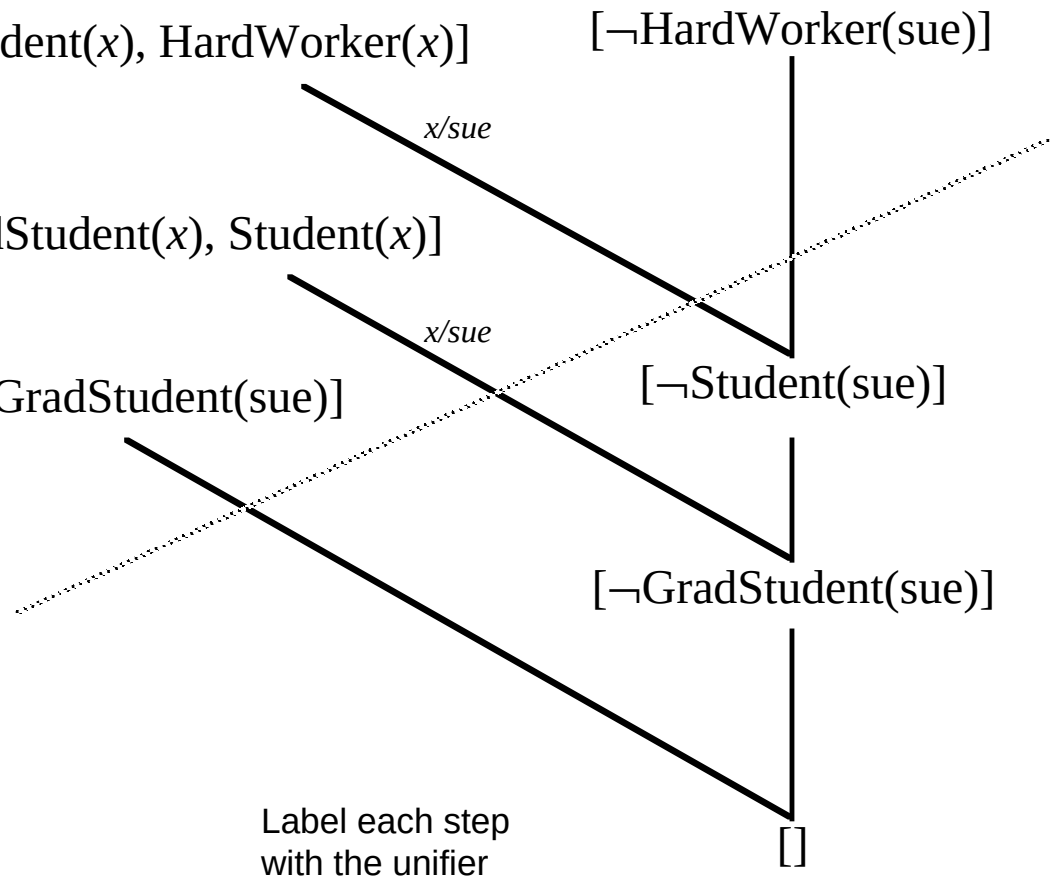
Theorem. $S \vdash ()$ iff S is unsatisfiable

A resolution example

1. $(P(x), Q(g(x)))$
2. $(R(a), Q(z), \neg P(a))$
3. R[1a,2c]{X=a} $(Q(g(a)), R(a), Q(z))$

- "R" means resolution step.
- "1a" means the 1st (a-th) literal in the first clause: $P(x)$.
- "2c" means the 3rd (c-th) literal in the second clause: $\neg P(a)$.
- 1a and 2c are the "clashing" literals.
- $\{X = a\}$ is the MGU applied.

Refutation example 1



?

KB |= HardWorker(sue)

KB

$\forall x$ GradStudent(x) $\supset$ Student(x)
$\forall x$ Student(x) $\supset$ HardWorker(x)
GradStudent(sue)

Label each step with the unifier

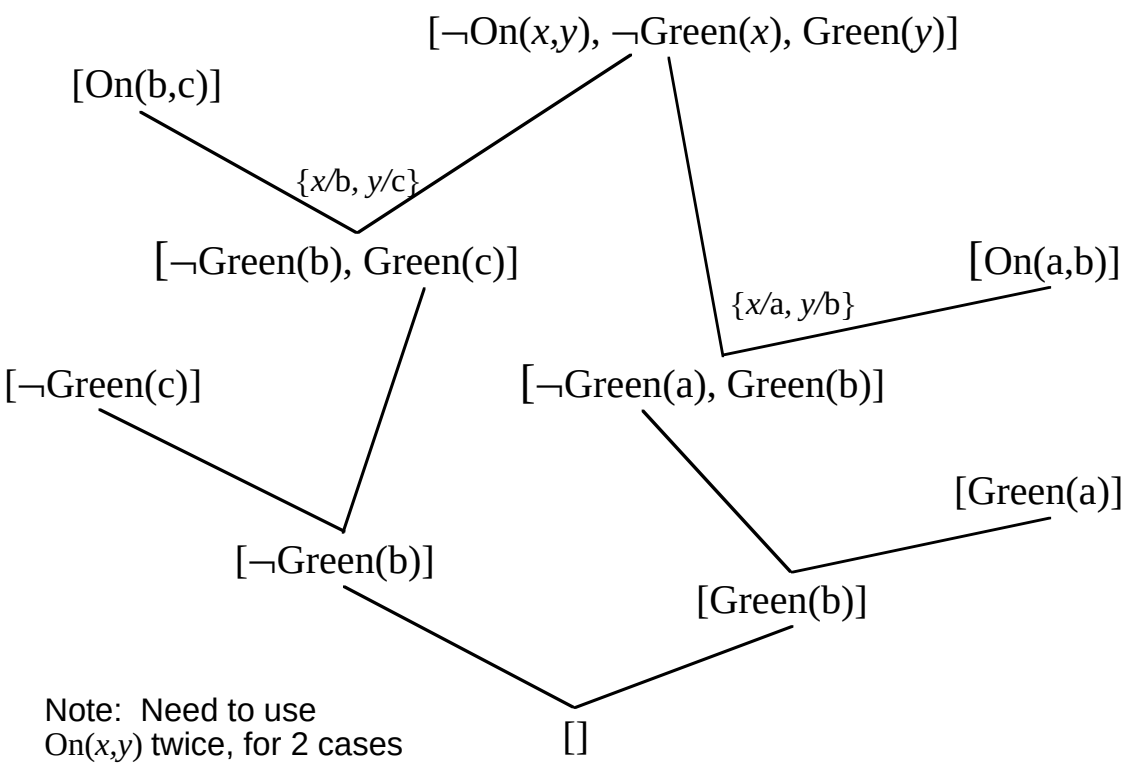
Point to relevant literals in clauses

The 3 blocks example

KB = {On(a,b), On(b,c), Green(a), ¬Green(c)} already in CNF

Query = $\exists x \exists y [On(x,y) \wedge Green(x) \wedge \neg Green(y)]$

Note: ¬Q has no existentials, so yields



Note: Need to use On(x,y) twice, for 2 cases

Alpine Club example

		1. $A(tony)$
		2. $A(mike)$
		3. $A(john)$
		4. $L(tony, rain)$
		5. $L(tony, snow)$
$\forall x(A(x) \wedge \neg S(x)) \rightarrow C(x)$	$\Rightarrow$	6. $(\neg A(x), S(x), C(x))$
$\forall x(C(x) \rightarrow \neg L(x, rain))$	$\Rightarrow$	7. $(\neg C(y), \neg L(y, rain))$
$\forall x(\neg L(x, snow) \rightarrow \neg S(x))$	$\Rightarrow$	8. $(L(z, snow), \neg S(z))$
$\forall x(L(tony, x) \rightarrow \neg L(mike, x))$	$\Rightarrow$	9. $(\neg L(tony, u), \neg L(mike, u))$
$\forall x(\neg L(tony, x) \rightarrow L(mike, x))$	$\Rightarrow$	10. $(L(tony, v), L(mike, v))$
$\neg\exists x(A(x) \wedge C(x) \wedge \neg S(x))$	$\Rightarrow$	11. $(\neg A(w), \neg C(w), S(w))$

Note that we must standardize variables.

Alpine Club example refutation

12. R[5, 9a]u = snow $\neg L(mike, snow)$
13. R[8,12]z = mike $\neg S(mike)$
14. R[6b, 13]x = mike $(\neg A(mike), C(mike))$
15. R[2,14a] $C(mike)$
16. R[8a, 12]z = mike $\neg S(mike)$
17. R[2,11]w=mike $(\neg C(mike), S(mike))$
18. R[15, 17] $S(mike)$
19. R[16,18] ()

Refutation examples

Prove that $\exists y \forall x P(x, y) \models \forall x \exists y P(x, y)$

- $\exists y \forall x P(x, y) \Rightarrow 1.P(x, a)$
- $\neg \forall x \exists y P(x, y) \Leftrightarrow \exists x \forall y \neg P(x, y) \Rightarrow 2.\neg P(b, y)$
- R[1,2]$\{x = b, y = a\}$()

$\neg \forall x (P(x) \vee Q(x))$
$\Rightarrow \exists x (\neg P(x) \wedge \neg Q(x))$
$\Rightarrow \neg P(a) \wedge \neg Q(a)$
$\Rightarrow 2.\neg P(a) \quad 3.\neg Q(a)$

1. $(P(x), Q(y))$

合取需要拆分成两个子句

Exercises: Prove

- $\forall x P(x) \vee \forall x Q(x) \models \forall x (P(x) \vee Q(x))$
- $\exists x (P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists x Q(x)$

R[1a,2]$\{x=a\}$ 4. $Q(y)$

R[4,3]$\{y=a\}$ ( )

Answer extraction

- We can also answer wh- questions
- Replace query $\exists x P(x)$ by $\exists x[P(x) \wedge \neg answer(x)]$
- Instead of deriving (), derive any clause containing just the answer predicate

加入~answer(x)是为了推导出结果x

推导出只包含答案的谓词子句

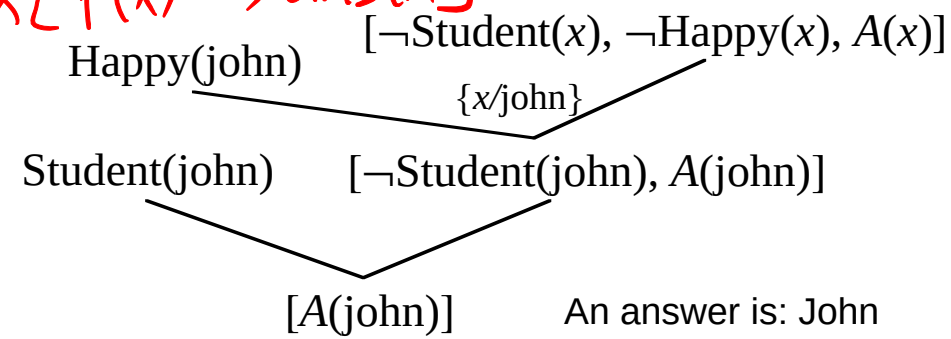
$\neg \exists x[P(x) \wedge \neg ans(x)]$

$\rightarrow \forall x[\neg P(x) \vee ans(x)]$

$\rightarrow \forall x[P(x) \rightarrow ans(x)]$

KB: Student(john)
Student(jane)
Happy(john)

Q: $\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



Alpine Club example answer extraction

- 11. $(\neg A(w), \neg C(w), S(w), answer(w))$
- The same resolution steps as before give us $answer(mike)$

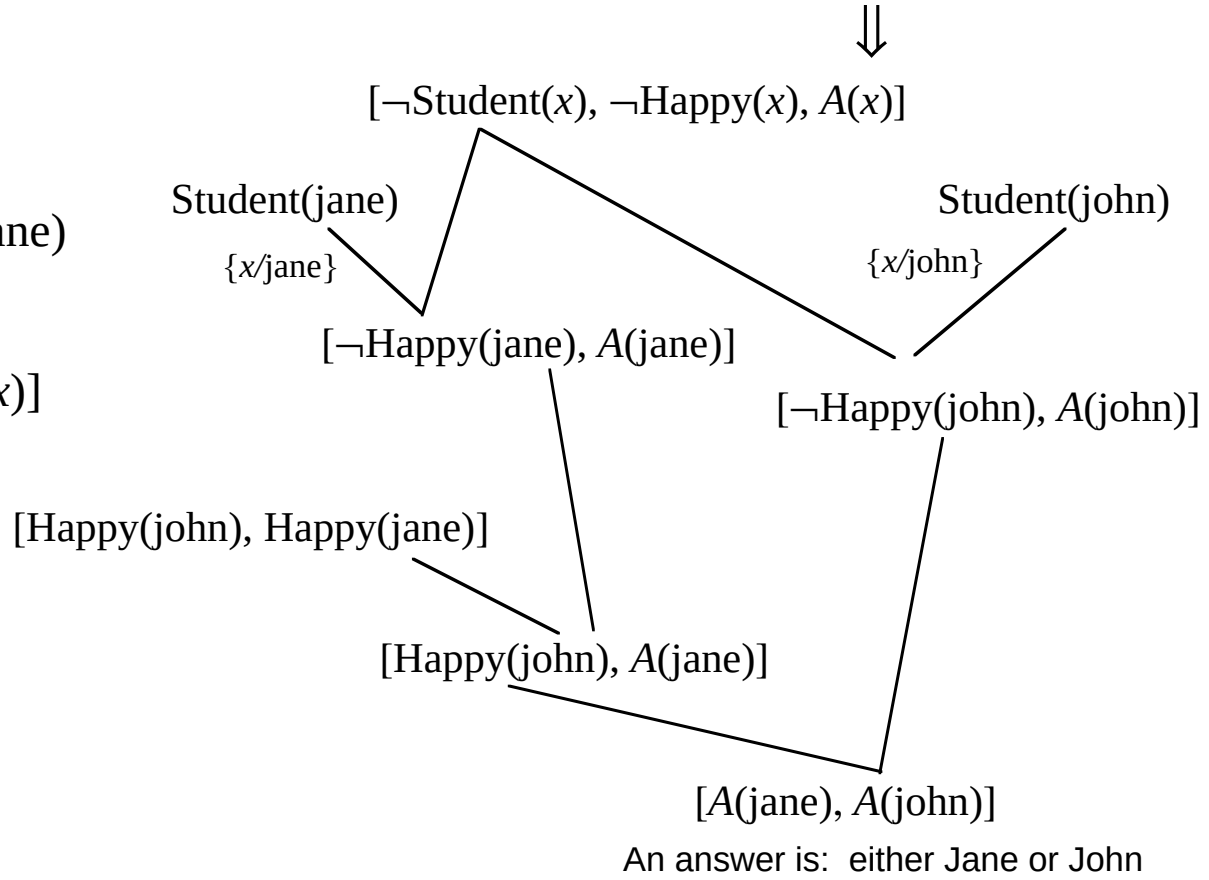
Disjunctive answers

KB:

Student(john)
Student(jane)
Happy(john) $\vee$ Happy(jane)

Query:

$\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



Note:

An answer is: either Jane or John

A problem

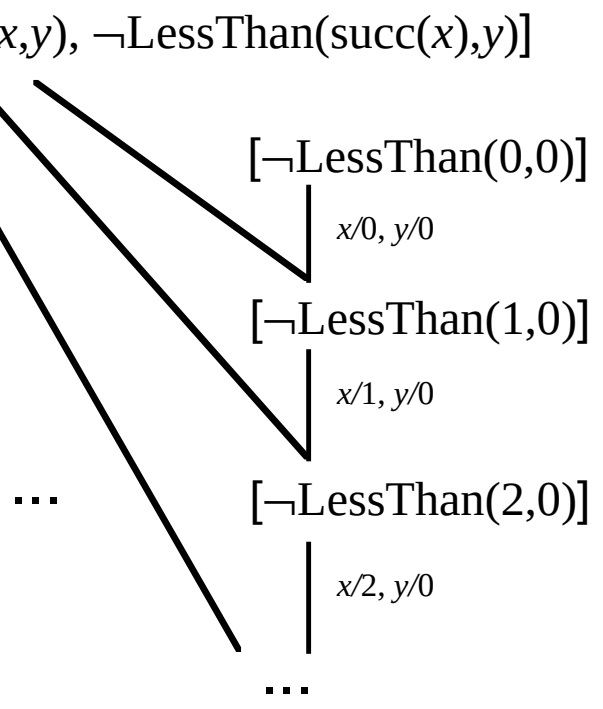
KB:

$\text{LessThan}(\text{succ}(x),y) \supset \text{LessThan}(x,y)$

Query:

$\text{LessThan}(\text{zero},\text{zero})$

Should fail since KB $\not\models$ Q



Infinite branch of resolvents

We use 0 for zero, 1 for succ(zero), 2 for succ(succ(zero)), . . .

Undecidability in the first-order case

不可判定性

- There can be no procedure to decide if a set of clauses is satisfiable.
- **Theorem.** $S \vdash ()$ iff S is unsatisfiable
- However, there is no procedure to check if $S \vdash ()$, because
- When S is satisfiable, the search for () may not terminate

Intractability in the propositional case

难处理性

- Determining if a set of clauses is satisfiable was shown by Cook in 1972 to be NP-complete.
- Satisfiability is believed by most people to be unsolvable in polynomial time.
- Procedures have been proposed (提出) for determining satisfiability that appear to work much better in practice than Resolution.
- They are called SAT solvers as they are mostly used to find a satisfying interpretation for clauses that are satisfiable.

Implications for KRR

启示

- In knowledge-based systems, actions depend on implicit beliefs, *i.e.*, logical entailments of KB
- However, as we have seen, computing entailments is unsolvable in general
- The hope is that in many practical scenarios, entailments can be efficiently computed
- In case entailments are difficult to compute, we seek for other ways out

Prolog and resolution

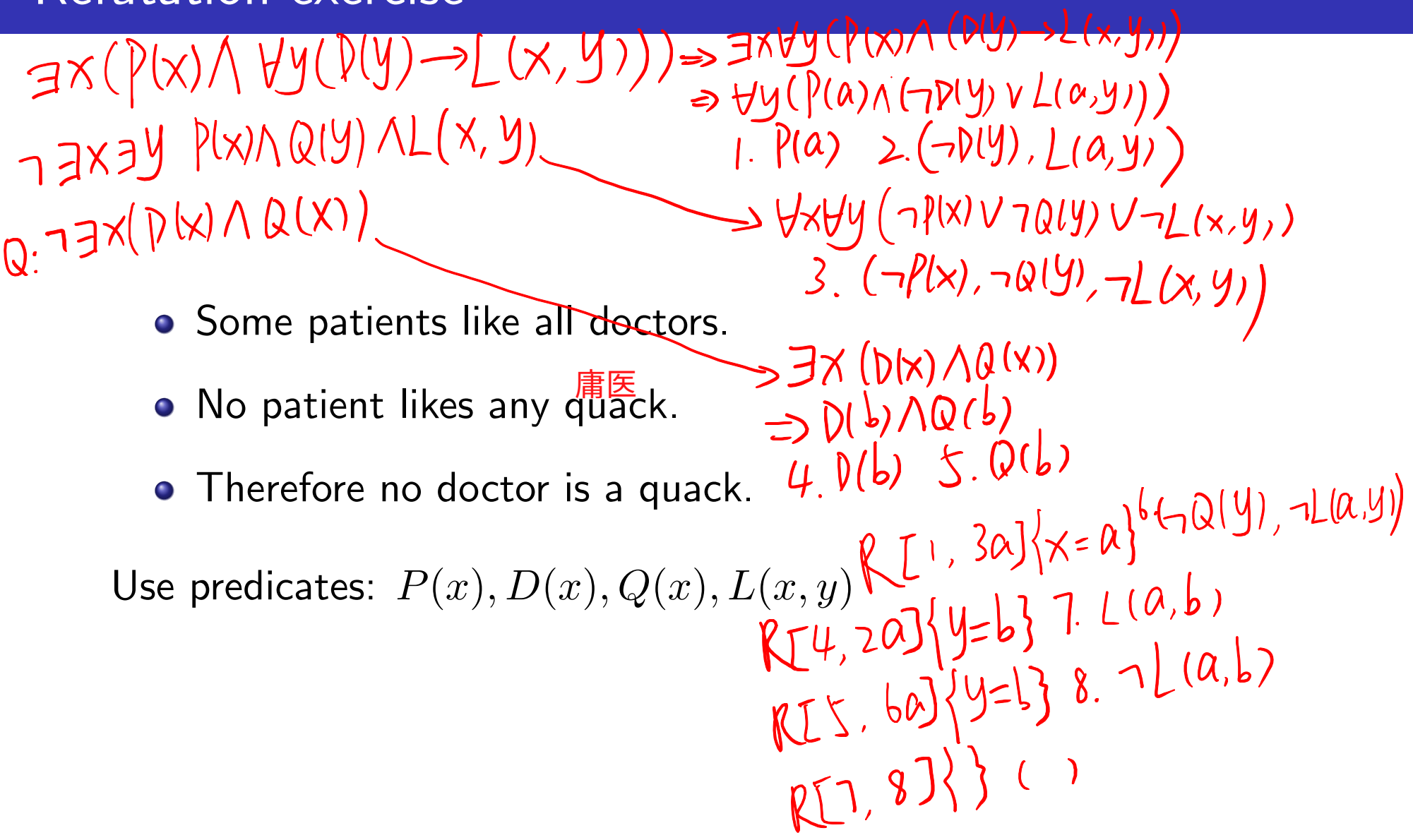
- Resolutions forms the basis of the implementation of Prolog
- When searching for (), Prolog uses a specific depth-first left-right strategy

Refutation exercise

- Some patients like all doctors.
- No patient likes any quack.
- Therefore no doctor is a quack.

庸医

Use predicates: $P(x), D(x), Q(x), L(x, y)$



Refutation exercise

- Whoever can read is literate. $\forall x(R(x) \to L(x))$
- Dolphins are not literate. $\forall x(D(x) \to \neg L(x))$
- Flipper is an intelligent dolphin. $D(F) \wedge I(F)$
- Who is intelligent but cannot read. $\exists x(I(x) \wedge \neg R(x) \wedge \neg \text{answers}(x))$

Use predicates: $R(x), L(x), D(x), I(x)$