Knowledge representation and reasoning (KRR)

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

What is KRR?

Symbolic encoding of propositions believed by some agent and their manipulation 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 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.
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
- Knowledge-based system: system with structures that
    - can be interpreted propositionally and
    - determine the system behavior

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

KRR and logic

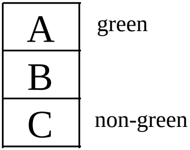
Logic is the main tool for KRR, because logic studies

- 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

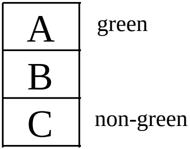
一阶逻辑

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

Alphabet

- Individuals (constants or 0-ary functions):
  - tony, mike, john
  - rain, snow

一元谓词

- 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): 二元
  - $L(x, y)$ 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):

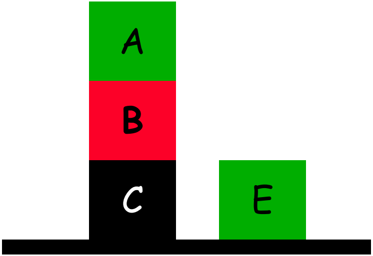
- Punctuation: 标点 (,),,,.
- Connectives and quantifiers: $=, \neg, \wedge, \vee, \forall, \exists$
- Variables: $x, x_1, x_2, \ldots, x', x'', \ldots, y, \ldots, z, \ldots$

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

- Predicate symbols
  - arity: number of arguments 参数数量
  - arity 0 predicates: propositional symbols 命题参数为0个
    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: 谓词
  - 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 原子公式
- 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

… (公式)+(谓词)+(公式)=公式

Notation

- Occasionally add or omit (,)
- Use [,] and $\{,\}$

简称

- Abbreviation: $(\alpha \rightarrow \beta)$ for $(\neg\alpha \vee \beta)$
- 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

变量域

- Free and bound occurrences of variables
- *e.g.*, $P(x) \wedge \exists x[P(x) \vee Q(x)]$
- A sentence: formula with no free variables
- 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 定义域
- 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.*, $I(P) \subseteq D^n$
    - If p is a 0-ary predicate symbol, *i.e.*, a propositional symbol, $I(p) \in \{true, false\}$ 对于任意一个命题，只有真或者假
- If f is a function symbol of arity n, $I(f)$ is an n-ary function over D, *i.e.*, $I(f) : D^n \to D$ 相当于变量的一组解
    - If c is a 0-ary function symbol, *i.e.*, a constant symbol, $I(c) \in D$

Blocks world example

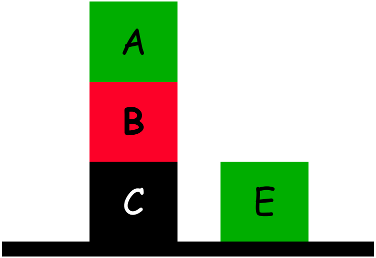
I: 谓词 → (变量)^n

- $D = \{\underline{A}, \underline{B}, \underline{C}, \underline{E}\}$

I

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

Environment



Denotation of terms

- Terms denote elements of the domain
- A variable assignment μ is a mapping from the set of variables to the domain D

给变量 v 赋值 μ

- $\|v\|_{\Im,\mu} = \mu(v)$
- $\|f(t_1, \ldots, t_n)\|_{\Im,\mu} = I(f)(\|t_1\|_{\Im,\mu}, \ldots, \|t_n\|_{\Im,\mu})$

Satisfaction: atomic formulas

$\models$ 满足

$\Im, \mu \models \alpha$ is read "$\Im, \mu$ satisfies α "

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

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

?

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

点不用管，相当于存在v，有α

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

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

无变量

与μ无关

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

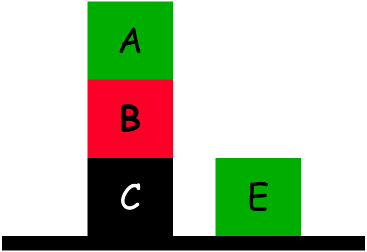
- Let S be a set of sentences
- $\Im \models S$, read $\Im$ satisfies S, if for every $\alpha \in \Im$, $\Im \models \alpha$
- If $\Im \models S$, we say $\Im$ is a model of S ?
- We say that S is satisfiable if there is $\Im$ s.t. $\Im \models S$, and
- e.g., is $\{\forall x(P(x) \to Q(x)), P(a), \neg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

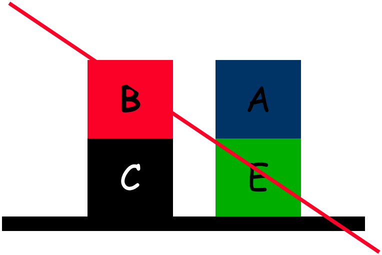
$(\forall x(\neg P(x) \vee Q(x)), P(a), \neg Q(a))$ X

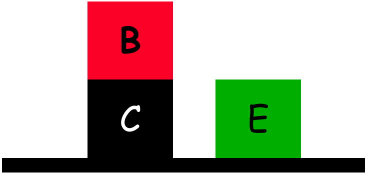
Blocks world example

KB

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



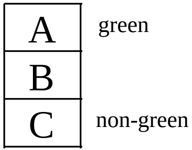




Logical entailment

- $S \models \alpha$ 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
- 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
- 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



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

Logical entailment: examples

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

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

给出一组解满足KB 但不满足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

隐式信念

- 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

让计算机可以检查是否KB逻辑蕴含a

- We want a mechanical procedure to check if $KB \models \alpha$
- Called an inference 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
- We begin with the propositional case
- Then proceed to the first-order case

从命题开始

一阶谓词

Clausal form

文字

- A literal is an atomic formula or its negation, *e.g.*, $p, \neg p$

  子句是文字的析取　可以用集合来表示
- A clause is a disjunction 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 $()$, representing false

  空子句为假，因为不可满足
- A formula is a conjunction of clauses, written as the set of clauses 公式是子句合取

formula = ( ) ∧ ( ) ∧ ..

Resolution rule of inference

(p, c_1) $(\neg p, c_2) \Rightarrow (c_1, c_2)$

$c_1 c_2$ 至少一个为真

- From the two clauses $\{p\} \cup c_1$ and $\{\neg p\} \cup c_2$, infer the clause $c_1 \cup c_2$
- $c_1 \cup c_2$ is called the resolvent of input clauses wrt the atom p
- *e.g.*, (p) and $(\neg p)$ resolve to $()$,
  (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: 两个子句的与 蕴含它们归结后的结果

Derivation

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

已有两个子句归结的结果

- 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

从S推出c

Soundness of derivations

如果S能推出c，则S蕴含c

- **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 $1 \leq i \leq n$, $S \models c_i$.
- However, the converse does not hold in general
  e.g., $(p) \models (p, q)$, but $(p) \not\vdash (p, q)$

  反过来不成立

Soundness and completeness of refutations

因为满足空子句代表为假

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

We will not prove the completeness part

Resolution-based inference procedure: refutation

如何推理 KB ⊢ α

if KB then α

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

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

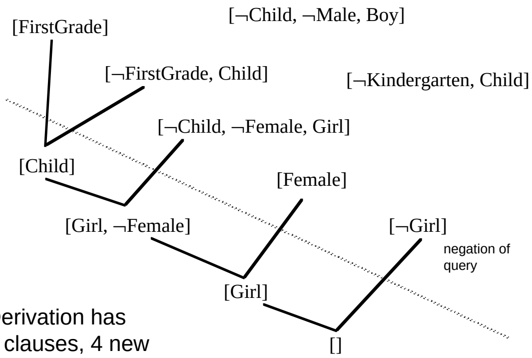
- put KB and $\neg\alpha$ into clausal form to get S,
- check if $S \vdash ()$

Refutation example 1

⊃是⊨

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

Show that KB $\models$ Girl



Derivation has
9 clauses, 4 new

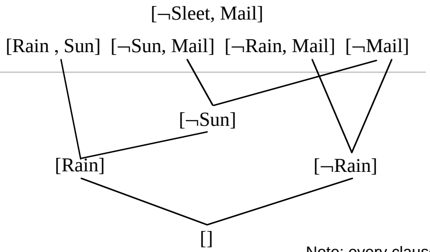
Refutation example 2

KB

(Rain ∨ Sun)
(Sun ⊃ Mail)
((Rain ∨ Sleet) ⊃ Mail)

$(\neg R \wedge \neg S) \vee M$
$= (\neg R, M)\ (\neg S, M)$

Show KB |= Mail



Note: every clause not in S has 2 parents

Similarly KB |≠ Rain

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

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

如何转化成子句

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, we invent a name for this individual, say a. This is a new constant symbol not equal to any previous constant symbols: 去掉存在量词
  $Elephant(a) \wedge Friendly(a)$ 用常数替代
- 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 $\forall x \exists y. Loves(x, y)$. y和x有关

- 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.
- 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.
- 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

引入一个函数符号h去掉存在量词

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

  w只和x和y有关，和z无关

A conversion example

删除蕴含

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

1. Eliminate implications using $A \to B \Leftrightarrow \neg A \lor B$

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

Move negations inwards

将非移入括号

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

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) 避免不同子句中的变量重复名字

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

Skolemize

引入函数符号g，代替存在z　去除存在量词

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

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)

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

将所有全称量词提到最前面

Disjunctions over conjunctions

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

6. Disjunctions over conjunctions using
$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$

子句里用析取，子句之间用合取

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

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).

去掉所有全称量词，每个变量都是代表任何一个和

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 合一

- Can the clauses $(P(john), Q(fred), R(x))$ and $(\neg P(y), R(susan), R(y))$ be resolved?
- Once reduced to clausal form, all remaining variables are universally quantified.
- So, implicitly the clause $(P(john), Q(fred), R(x))$ represents $(P(john), Q(fred), R(john))$, $(P(john), Q(fred), R(fred))$, $\ldots$
- So there is a specialization of $(P(john), Q(fred), R(x))$ that can be resolved with a specialization of $(\neg P(y), R(susan), R(y))$ 对这两条式子取一个特例，能否归并起来
- 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

冲突的文字

- 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!

不希望太过特殊化

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

用来找最一般的归一结果

- Unification is a mechanism for finding a most general matching
- A key component of unification is substitution. 替换
- 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). 如 $a = b$
- 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 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 $f(a)$.

Composition of substitutions

消替换组合

- We can compose two substitutions θ and σ to obtain a new substitution $\theta\sigma$
- Let $\theta = \{x_1 = s_1, x_2 = s_2, \ldots, x_m = s_m\}$,
  $\sigma = \{y_1 = t_1, y_2 = t_2, \ldots, y_k = t_k\}$

  相当于代入了sigma
- 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 $V = V$.

  自己等于自己的删掉
- Step 3. Delete any equation $y_i = s_i$ where y_i is equal to one of the x_j in θ. Why? 在后面的重复被赋值的删掉

Composition example

替换组合举例

- 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 $y = y$.
- Step 3. Delete $x = a$.
- The result is $S = \{x = f(b), y = b, z = y\}$

Note on substitutions

空替换 f不变

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

Unifiers

- A unifier of two formulas f and g is a substitution σ that makes f and g syntactically identical. 找到一个代入让这两个公式等价
- 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.

a和w是常量

MGU

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

首先是个归一

- σ is a unifier.

sigma是最一般的代入，还可以代入别的组成别的代入

- For every other unifier θ of f and g there must exist a third substitution λ such that $\theta = \sigma\lambda$. 且可以组成任意其它归一

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.

如果两个句子可以合一，那么他们的MGU一定是唯一的

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 $\lambda = \{x = a\}$

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. $k = 0$; $\sigma_0 = \{\}$; $S_0 = \{f, g\}$

   如果两个公式相同，则返回

2. If S_k contains an identical pair of formulas, stop and return σ_k as the MGU of f and g.

   找出不同的集合对

3. Else find the disagreement set $D_k = \{e_1, e_2\}$ of S_k

   找到 $V=t$ 使 e_1, e_2 匹配

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. 找不到则返回

Computing MGU examples

注意，项是不能赋值的，比如不能
g(x)=a是不行的，只能代入一个变量

不行

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 $P(y, f(y))$

$\sigma = \{x := y,\ y := f(y)\}$

不行，因为 $f(y)$ 含有 y

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

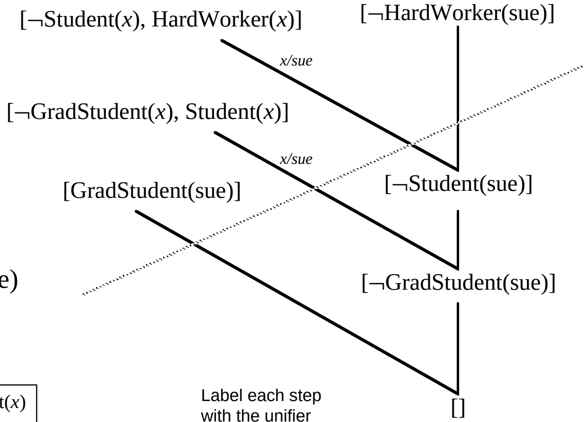
A resolution example

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

↖归结

- "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

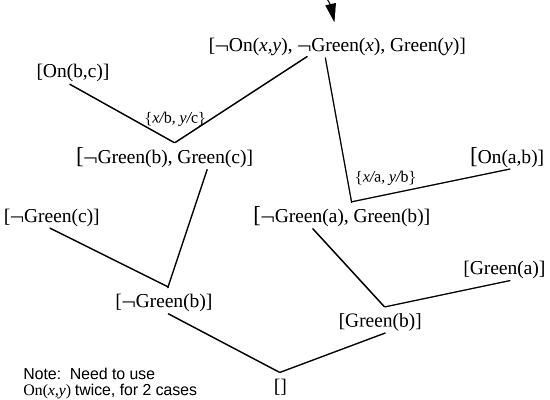
Point to relevant
literals in clauses

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



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] ()

推出空子句
则逻辑上蕴含

Refutation examples

$\forall x(\neg P(x) \wedge \neg Q(x))$

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

- $\exists y \forall x P(x,y) \Rightarrow 1.P(x,a)$ 存在用特定常量a代替
- $\neg \forall x \exists y P(x,y) \Leftrightarrow \exists x \forall y \neg P(x,y) \Rightarrow 2.\neg P(b,y)$
- R[1,2]$\{x=b, y=a\}()$

$\neg \forall x(P(x) \vee Q(x))$

Exercises: Prove

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

- $\forall x P(x) \vee \forall x Q(x) \models \forall x(P(x) \vee Q(x))$ 2 $\neg P(a)$ 3 $\neg Q(a)$
- $\exists x(P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists x Q(x)$

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

5. R[3, 4]$(y=a)$ ()

Answer extraction

$\forall x \neg(P(x) \wedge \neg A(x))$
$\Rightarrow \forall x(P(x) \rightarrow A(x))$

- We can also answer wh- questions

查询变为求答案

?

- Replace query $\exists x P(x)$ by $\exists x[P(x) \wedge \neg answer(x)]$ 加上这一项
- Instead of deriving (), derive any clause containing just the answer predicate

加上一项

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

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

Happy(john) $\quad$ [$\neg$Student(x), $\neg$Happy(x), A(x)]

{x/john}

Student(john) $\quad$ [$\neg$Student(john), A(john)]

[A(john)] $\quad$ An answer is: John

Alpine Club example answer extraction

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

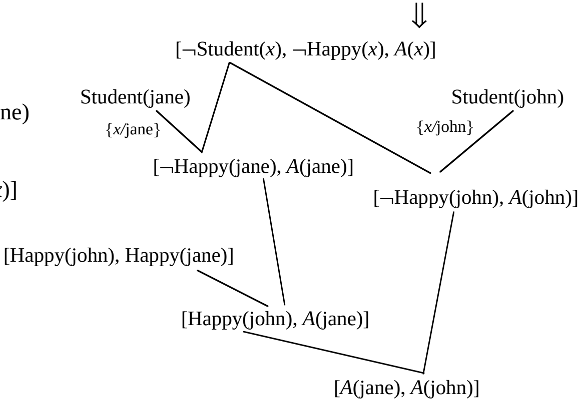
Disjunctive answers

KB:

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

Query:

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



An answer is: either Jane or John

Note:

A problem

无法证明这个知识库
是否都为真

[LessThan(x,y), ¬LessThan(succ(x),y)]

[¬LessThan(0,0)]

x/0, y/0

[¬LessThan(1,0)]

x/1, y/0

…

[¬LessThan(2,0)]

x/2, y/0

…

KB:

LessThan(succ(x),y) $\supset$ LessThan(x,y)

Query:

LessThan(zero,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

- There can be no procedure to decide if a set of clauses is satisfiable. 但无法判定是否都为真
- **Theorem.** $S \vdash ()$ iff S is unsatisfiable 只能判断是否不可满足
- However, there is no procedure to check if $S \vdash ()$, because (加个反例证明不可满足)
- When S is satisfiable, the search for $()$ may not terminate

Intractability in the propositional case

- 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 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

- 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

$\exists x (D(x) \rightarrow Q(x))$

$\underline{\exists x (D(x) \wedge Q(x))}$

- Some patients like all doctors. $\exists x (P(x) \wedge \forall y D(y) \rightarrow L(x, y))$
- No patient likes any quack. $\neg \exists x \exists y [P(x) \wedge Q(y) \wedge L(x, y)]$
- Therefore no doctor is a quack. $\neg \exists x (D(x) \wedge Q(x))$

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

1. $P(a)$
2. $D(y)$
3. $L(a, y)$

$\exists x D(x) \rightarrow Q(x)$

$\exists x D(x) \wedge Q(x)$

Refutation exercise

- Whoever can read is literate.
- Dolphins are not literate.
- Flipper is an intelligent dolphin.
- Who is intelligent but cannot read.

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