Conversion to Clausal Form: 8 steps

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

Skolemization

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:
  $Elephant(a) \wedge Friendly(a)$
- 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.
- 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

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

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 $A \rightarrow B \Leftrightarrow \neg A \vee B$

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

Move negations inwards

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

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)

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

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)

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

注意：教材中先求前束范式再 Skolem 化，这样引入的函数可能更复杂，比如这里是 $g(x,y)$

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

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

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).

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?
- Once reduced to clausal form, all remaining variables are universally quantified.
  $(P(john), Q(fred), R(john))$, $(P(john), Q(fred), R(fred))$, $\ldots$
- So there is a specialization of $(P(john), Q(fred), R(x))$ that can be resolved with a specialization of $(\neg P(y), R(susan), R(y))$
- 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

- 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.
- To define the most-general unifier, we need to define substitutions.

Substitution (置换)

- A key component of unification is substitution.
- 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 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 (置换的复合)

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

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 $y = y$.
- Step 3. Delete $x = a$.
- 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 (结合的): $(f\theta)\sigma = f(\theta\sigma)$
- Composition is simply a way of converting the sequential application of a series of substitutions to a single substitution.

Unifiers

- A unifier (合一项) 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.

MGU

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

- 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.

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 (对齐) 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 (差异集).
- The algorithm works by successively fixing disagreement sets (逐项修正差异集) until the two formulas become syntactically identical.

Computing MGUs

Given two atomic formulas f and g

1. $\sigma = \{\}$; $S = \{f, g\}$
2. If S contains an identical pair of formulas, stop and return σ as the MGU of f and g.
3. Else find the disagreement set $D = \{e_1, e_2\}$ of S
4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let $\sigma = \sigma\{V = t\}$; $S = S\{V = t\}$; Goto 2
5. Else stop, f and g cannot be unified.

Note: to update σ, we must compose σ with $\{V = t\}$.
A common error is to just add $V = t$ to σ.

Computing MGU examples

1. $P(f(a), g(x))$ and $P(y, y)$: un-unifiable
2. $P(a, x, h(g(z)))$ and $P(z, h(y), h(y))$
   MGU: $\{z = a, x = h(g(a)), y = g(a)\}$
3. $P(x, x)$ and $P(y, f(y))$: un-unifiable

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 $(c_1 \cup c_2)\sigma$

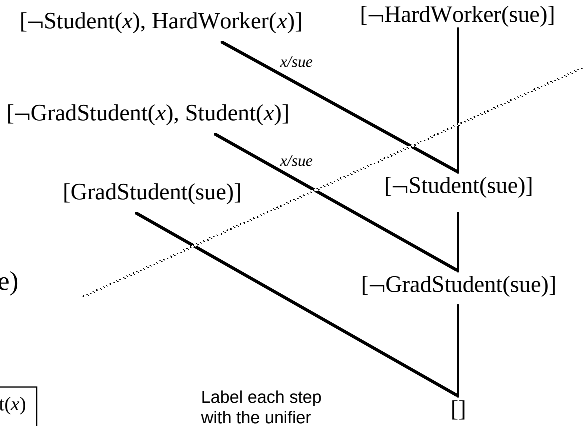
Theorem. $S \vdash ()$ iff S is unsatisfiable

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 (冲突文字).
- $\{X = a\}$ is the MGU applied.

Refutation example 1



$$\mathrm{KB} \overset{?}{\models} \mathrm{HardWorker}(sue)$$

KB

$\forall x\ \mathrm{GradStudent}(x) \supset \mathrm{Student}(x)$
$\forall x\ \mathrm{Student}(x) \supset \mathrm{HardWorker}(x)$
$\mathrm{GradStudent}(sue)$

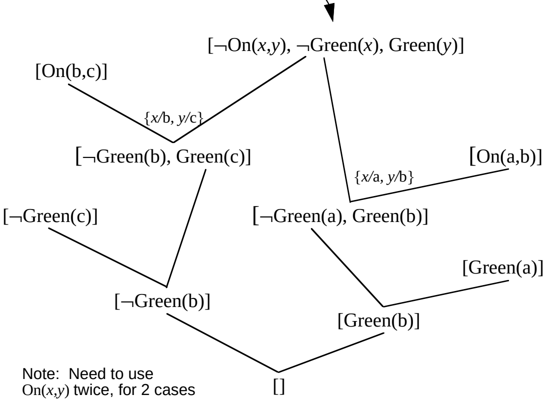
Label each step with the unifier

Point to relevant literals in clauses

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. $L(tony, snow)$

$$
\begin{array}{lcl}
\forall x(A(x) \land \neg S(x)) \to C(x) & \Rightarrow & 6.\ (\neg A(x), S(x), C(x)) \\
\forall x(C(x) \to \neg L(x, rain)) & \Rightarrow & 7.\ (\neg C(y), \neg L(y, rain)) \\
\forall x(\neg L(x, snow) \to \neg S(x)) & \Rightarrow & 8.\ (L(z, snow), \neg S(z)) \\
\forall x(L(tony, x) \to \neg L(mike, x)) & \Rightarrow & 9.\ (\neg L(tony, u), \neg L(mike, u)) \\
\forall x(\neg L(tony, x) \to L(mike, x)) & \Rightarrow & 10.\ (L(tony, v), L(mike, v)) \\
\neg \exists x(A(x) \land C(x) \land \neg S(x)) & \Rightarrow & 11.\ (\neg A(w), \neg C(w), S(w))
\end{array}
$$

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]$\{x = b, y = a\}()$

Exercises: Prove

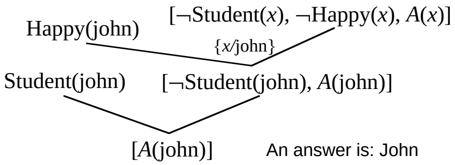
- $\forall x P(x) \vee \forall x Q(x) \models \forall x (P(x) \vee Q(x))$
- $\exists x (P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists x Q(x)$

Answer extraction (答案抽取)

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

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

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



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

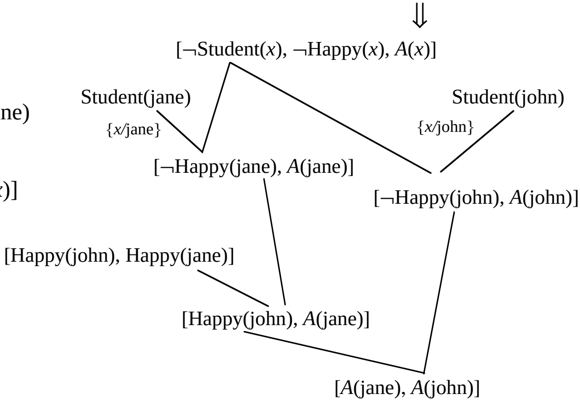
Disjunctive answers

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

以下 4 面为扩展内容

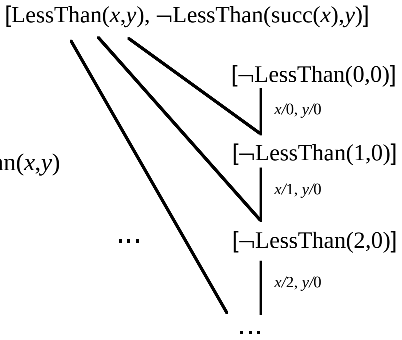
KB:

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.
- **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

- 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

- Prolog is a language that is useful for doing symbolic and logic-based computation
- Resolutions forms the basis of the implementation of Prolog
- When searching for (), Prolog uses a specific depth-first left-right strategy

王浩：机器定理证明的奠基人

- 王浩（1921 - 1995），美籍华裔哲学家、数理逻辑学家。
- 1921 年出生在山东济南，1943 年西南联合大学数学系毕业，1945 年清华大学哲学系毕业，师从著名逻辑学家金岳霖。
- 1948 年哈佛大学逻辑学博士毕业，成为哈佛的助理教授。
- 1956–1961 年任牛津大学数学哲学高级讲师。
- 1961–1967 年回到哈佛任数理逻辑与应用数学教授。
- 他在 1958 年夏天写的程序在 IBM-704 上, 只用九分钟就证明了罗素《数学原理》中一阶逻辑的全部定理。
- 在 1983 年于国际人工智能联合会议荣获首届证明自动化里程碑奖（the first Milestone Prize for Automated Theorem-Proving）

Refutation exercise

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

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

Refutation exercise

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

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