一阶逻辑推理

1 代换、全称量词实例化、存在量词实例化
2 合一、CNF、归结推理
3 前向链接与后向链接

New Quantifiers in FOL (review)

- Universal quantifier(全称量词) ∀
  - ∀x means “For all x…”
  - Always true
  - Usually used with ⇒
  - ¬∀x means “Not all x…”
- Existential quantifier (存在量词) ∃
  - ∃x means “There exists an x…”
  - True for at least one interpretation
  - Usually used with ∧
  - ¬∃x means “There exists no x…”

Sentences in FOL (review)

- Term (项)
  - Constant symbol, variable symbol, or function symbol
- atomic sentence(原子语句)
  - Predicate symbol with value true or false
  - Represents a relation between terms
- complex sentence(复合语句)
  - Atom(s) joined together using logical connectives and/or quantifiers

Concepts （ review ）

- Literal(文字)
  - 原子、原子的否定
- Clause and Clause set
  - disjunctions of literals(文字的析取)
  - e.g. $P(x) \vee \neg Q(x,y)$, $\neg P(x,c) \vee R(x,y,f(x))$
- CNF(Conjunctive normal form, 合取范式)
  - conjunction of disjunctions of literals
  - CNFs do not contain quantifiers!!!

1 代换、量词实例化

- 已知命题逻辑的推理策略，
  一阶逻辑下如何推理？

Logics in General

- What exists in the world — TRUTH
  - Propositional Logic: facts hold or do not hold.
  - First Order Logic: objects with relations between them that hold or do not hold

- *First order inference* can be done by converting the knowledge base to PL and using *propositional inference*.
  - How to convert universal quantifiers?
  - How to convert existential quantifiers?

Barber Paradox

- The barber shave all those men, and only those men, who do not shave themselves.
- There is no barber in town.

- B(x): x is a barber.
- H(x,y): x shave y.

FOL sentences:

- ________________________________
- ________________________________
- ________________________________
- how to prove it using resolution like in PL?

Barber Paradox

- The barber shave all those men, and only those men, who do not shave themselves.
- There is no barber in town.

- B(x): x is a barber.
- H(x,y): x shave y.

FOL sentences:

- $(\forall x)(B(x) \Rightarrow (\forall y)(\neg H(y,y) \Rightarrow H(x,y)))$
- $(\forall x)(B(x) \Rightarrow (\forall y)(H(y,y) \Rightarrow \neg H(x,y)))$
- $\neg(\exists x)B(x)$
- how to prove it using resolution like in PL?
  - convert to CNF, which contains no quantifiers, and then use resolution rules.

Substitution {9.1.1}

- P∨R, Q∨¬R can infer P∨Q using resolution in PL.
- P∨R(a), Q∨¬R(a) can infer P∨Q
- can P∨R(x), Q∨¬R(y) infer P∨Q ?

- First, we need substitution

Substitution {9.1.1}

- FOL is Similar to PL
  - Important differences
    - Quantifiers
    - Variables

substitution

sentence

- Important concept: *substitution*(代换)
  - Subst(θ, α), θ is like {x/Michael, y/Bob}
  - replace variables with terms

$$(\forall x)(Man(x) \Rightarrow Mortal(x))$$

$$\text{Subst}(\{x / Michael\}, (\forall x)(Man(x) \Rightarrow Mortal(x)))$$

$$Man(Michael) \Rightarrow Mortal(Michael)$$

Substitution {9.1.1}

- Subst({Michael/x}, Man(Michael) ⇒ Mortal(Michael))
  = Max(x) ⇒ Mortal(x)   correct??
- Subst({Michael/Bob}, Man(Michael) ⇒Mortal(Michael))
  = Max(Bob) ⇒ Mortal(Bob)   correct??
- Subst({x/Bob}, Man(x) ⇒ Mortal(x))
  = Max(Bob) ⇒ Mortal(Bob)   correct??

Substitution {9.1.1}

- Subst({Michael/x}, Man(Michael) ⇒ Mortal(Michael))
  = Max(x) ⇒ Mortal(x) no

- Subst({Michael/Bob}, Man(Michael) ⇒Mortal(Michael))
  = Max(Bob) ⇒ Mortal(Bob) no

- Subst({x/Bob}, Man(x) ⇒ Mortal(x))
  = Max(Bob) ⇒ Mortal(Bob) yes

inference rules for quantifiers{9.1.1}

$$\frac{\forall v \quad \alpha}{\textsc{Subst}(\{v/g\},\alpha)}$$

- Universal Instantiation (实例化)
- Everyone at AI class is smart.
  - (∀x) (AtAI(x) ⇒ Smart(x))
  - after substitution (代换) {x/张三}, {x/李四}, {x/brother(王五)} becomes
  - AtAI(张三) ⇒ Smart(张三)
  - AtAI(李四) ⇒ Smart(李四)
  - AtAI(brother(王五)) ⇒ Smart(brother(王五))

- We've replaced the variable with ***all possible*** ground terms (基项,terms without variables)

inference rules for quantifiers{9.1.1}

- Existential Instantiation (实例化)
- Example: Someone at AI is sleeping in class. (∃x)(AtAI(x)∧Sleep(x))
  - Let's call it *a*
  - becomes: (AtAI(*a*)∧Sleep(*a*))

$$\frac{\exists v \quad \alpha}{\textsc{Subst}(\{v/k\},\alpha)}$$

- You can replace the variable with **a *constant symbol* that does not appear elsewhere** in the knowledge base
- The constant symbol is a *Skolem* constant

inference rules for quantifiers{9.1.1}

- Existential instantiation
  - consider: Everyone has a mother.
  - $\forall y\ \exists x\ Mother(x,y)$
  - how to instantiate x?
  - $\forall y\ Mother(M_{12},y)$ correct?
    M_{12} a *constant symbol* that does not appear elsewhere in the knowledge base

inference rules for quantifiers{9.1.1}

- Existential instantiation
  - consider: Everyone has a mother.
  - $\forall y\ \exists x$ Mother(x,y)

  - how to instantiate x?
  - $\forall y$ Mother(m(y),y)
  - m(y) is a Skolem function

inference rules for quantifiers{9.1.1}

- Instantiate

$(\forall x_1)\ldots(\forall x_{k-1})(\exists x_k)(Qx_{k+1})\ldots(Qx_n)M(x_1, \ldots, x_k, \ldots, x_n)$

replace x_k with a Skolem function $f(x_1, \ldots, x_{k-1})$

$(\forall x_1)\ldots(\forall x_{k-1})(Qx_{k+1})\ldots(Qx_n)M(x_1,\ldots,f(x_1,\ldots,x_{k-1}),\ldots,x_n)$

inference rules for quantifiers{9.1.1}

- Only perform substitution once for existential quantifier
  - ∃x Kill (x, Victim)
  - There exists a *x* who killed *Victim*.
    - Someone killed the victim
    - Maybe more than one person killed the victim
    - Existential quantifier says at least one person was killer
  - Replacement is
    - Kill (Murderer, Victim)

合一、CNF、归结

Substitution {9.1.1}

- $P \vee R$, $Q \vee \neg R$ can infer $P \vee Q$ using resolution in PL.

- can $P \vee R(x)$, $Q \vee \neg R(y)$ infer $P \vee Q$?

- First, we need substitution
- what is the substitution for this problem?

Substitution {9.1.1}

- $P \vee R$, $Q \vee \neg R$ can infer $P \vee Q$ using resolution in PL.
- can $P \vee R(x)$, $Q \vee \neg R(y)$ infer $P \vee Q$?
- First, we need substitution
- we need a **unifier** *θ*
  where SUBST(θ,R(x))=SUBST(θ,R(y))
- *a unifier is a substitution*

Unifier {9.2.2}

- Unification(合一) is the process of finding substitutions
- *UNIFY* takes two sentences and returns a unifier(合一元) if one exists

UNIFY(p,q)=θ where SUBST(θ,p)=SUBST(θ,q)

Unifier {9.2.2}

- We can get the inference immediately if we can find a substitution.

UNIFY(p,q)=θ where SUBST(θ,p)=SUBST(θ,q)

α	β	θ
P(x)	P(a)	{x/a}
P(f(x),y,g(y))	P(f(x),x,g(x))	{x/y}
P(f(x),z,z)	P(f(x),g(a,y),g(a,y))	{z/g(a,y)}

Unifier {9.2.2}

- Multiple unifiers are possible
  - UNIFY (Knows(John,x), Knows(y,z))
  - {y/John, x/z} or {x/John, y/John, z/John}
  - Knows (John, z) or Knows (John, John)
  - First unifier is more general than second because it places fewer restrictions on the values of the variables
- There is a *most general unifier (MGU)* for every unifiable pair of expressions

Unifier {9.2.2}

- Consider the sentence
  - UNIFY(Knows(John,x),Knows(x,Elizabeth)) =______________________________

Unifier {9.2.2}

- Consider the sentence
  - UNIFY(Knows(John,x),Knows(x,Elizabeth)) = Fail
    - This fails because *x* cannot take on two values
    - But "Everyone knows Elizabeth" and it should not fail
  - How?

Unifier {9.2.2}

- Consider the sentence
  - UNIFY(Knows(John,x),Knows(x,Elizabeth)) = Fail
    - This fails because *x* cannot take on two values
    - But "Everyone knows Elizabeth" and it should not fail
  - Must **standardize** apart one of the two sentences to eliminate reuse of variable
    - UNIFY (Knows(John,x), Knows(x1,Elizabeth))

Convert to CNF {9.5.1}

- Essentially same as propositional logic
  - Eliminate implications
  - Move negation inwards
  - Standardize variables
  - Skolemize
  - Drop universal quantifiers
  - Distribute $\vee$ over $\wedge$

- Split Conjunctions into clauses
- try:

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

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

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

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

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

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

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

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

CNF

SPLIT CNF into clauses:

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

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

$$\neg P(x)\vee\neg P(g(x))$$

Resolution: PL version {9.5.2}

- in propositional logic:
- Resolution inference rule (for CNF)

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_i and m_j are complementary literals.

- Resolution Refutation(归结反驳)
  - Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$) and infer contradiction (empty clause)

Resolution: FOL version {9.5.2}

- in first-order logic:
- Resolution inference rule (for CNF)

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\text{SUBST}(\theta, \ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

where UNIFY(ℓ_i, $\neg m_j$)= θ .

- Resolution Refutation(归结反驳)
  - Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$) and infer contradiction (empty clause)

Examples {9.5.2}

E1	1)	$B(x)$	
	2)	$\neg B(x) \lor C(x)$	
	3)	$C(x)$	1)和2)
E2	1)	$P(x) \lor Q(x)$	
	2)	$\neg Q(f(y))$	
	3)	$P(f(y))$	1)和2) $\{x/f(y)\}$
E3	1)	$P(x,f(a)) \lor P(x,f(y)) \lor Q(y)$	
	2)	$\neg P(z,f(a)) \lor \neg Q(z)$	
	3)	$P(z, f(y)) \lor Q(y) \lor \neg Q(z)$	1)和2) $\{x/z\}$
	4)	$P(x, f(a)) \lor P(x, f(y)) \lor \neg P(y, f(a))$	1)和2) $\{z/y\}$
	5)	$Q(a) \lor \neg Q(z)$	1)和2) $\{x/z, y/a\}$

Resolution Steps {9.5.3}

1. Convert problem into FOL KB
2. Convert FOL statements in KB to CNF
3. Add negation of query (in CNF) in KB
4. Use resolution to infer new clauses from KB
5. Produce a contradiction that proves our query

{9.5.3}

exercise1 ：

- $(\forall x)(\forall y)\neg P(x, y)$,
  $(\forall x)(\forall y)(Q(x, y) \Rightarrow P(y, x)) \models$ **$\neg (\forall x)\, Q(x, a)$**

1) $\neg P(x, y)$
2) $\neg Q(x, y) \vee P(y, x)$
3) $Q(x, a)$
4) $P(a, x)$ 2)和3) y/a
5) NIL 1)和4) $x/a, y/a$

{9.5.3}

exercise2:

$\{\neg P(x) \vee P(f(x)), P(a), \neg P(f(f(f(a))))\}$ is unsatifiable

1) $\neg P(x) \vee P(f(x))$	
2) $P(a)$	
3) $\neg P(f(f(f(a))))$	
4) $\neg P(f(f(a)))$	1) 和3) $x/f(f(a))$
5) $\neg P(f(a))$	1) 和4) $x/f(a)$
6) $\neg P(a)$	1) 和5) x/a
7) NIL	2) 和6)

Barber Paradox{9.5.3}

- exercise3: The barber shave all those men, and only those men, who do not shave themselves.
- There is no barber in town.
- B(x): x is a barber.
- H(x,y): x shave y.

FOL sentence:

- $(\forall x)(B(x) \Rightarrow (\forall y)(\neg H(y,y) \Rightarrow H(x,y)))$
- $(\forall x)(B(x) \Rightarrow (\forall y)(H(y,y) \Rightarrow \neg H(x,y)))$
- $\neg(\exists x)B(x)$

1) $\neg B(x) \vee H(x, y) \vee H(y, y)$
2) $\neg B(x) \vee \neg H(x, y) \vee \neg H(y, y)$
3) $B(b)$
4) $\neg B(x)$　　1) 和2) y/x
5) NIL　　3) 和4) b/x

Quack problem{9.5.3}

exercise4 :

- Some patients like every doctor, but nobody likes quack. So, there is no quack.
- Predicates: P(x), D(x), Q(x), L(x,y)

F1 : $(\exists x)(P(x)\wedge(\forall y)(D(y) \Rightarrow L(x,y)))$

F2 : $(\forall x)(P(x) \Rightarrow (\forall y)(Q(y) \Rightarrow \neg L(x,y)))$

G : $(\forall x)(D(x) \Rightarrow \neg Q(x)$

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

proof : F1,F2 $\models$ G

Quack problem{9.5.3}

F1 : $(\exists x)(P(x) \wedge (\forall y)(D(y) \Rightarrow L(x,y)))$

$(\exists x)(P(x) \wedge (\forall y)(\neg D(y) \vee L(x,y)))$

$P(a) \wedge (\forall y)(\neg D(y) \vee L(a,y))$

$(\forall y)(P(a) \wedge (\neg D(y) \vee L(a,y)))$

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

F2: $(\forall x)(P(x) \Rightarrow (\forall y)(Q(y) \Rightarrow \neg L(x,y)))$

$(\forall x)(\neg P(x) \vee (\forall y)(\neg Q(y) \vee \neg L(x,y)))$

$(\forall x)(\forall y)\ (\neg P(x) \vee (\neg Q(y) \vee \neg L(x,y)))$

$\neg P(x) \vee \neg Q(z) \vee \neg L(x,z)$

resolution: quack problem{9.5.3}

G: $(\forall x)(D(x) \Rightarrow \neg Q(x))$

$\neg$ G: $\neg(\forall x)(D(x) \Rightarrow \neg Q(x))$

$\equiv \neg(\forall x)(\neg D(x) \vee \neg Q(x))$

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

1) $D(b)$
2) $Q(b)$
3) $P(a)$
4) $\neg D(y) \vee L(a,y)$
5) $\neg P(x) \vee \neg Q(z) \vee \neg L(x,z)$

F1: $(\exists x)(P(x) \wedge (\forall y)(D(y) \Rightarrow L(x,y)))$

$(\exists x)(P(x) \wedge (\forall y)(\neg D(y) \vee L(x,y)))$

$P(a) \wedge (\forall y)(\neg D(y) \vee L(a,y))$

$(\forall y)(P(a) \wedge (\neg D(y) \vee L(a,y)))$

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

F2: $(\forall x)(P(x) \Rightarrow (\forall y)(Q(y) \Rightarrow \neg L(x,y)))$

$(\forall x)(\neg P(x) \vee (\forall y)(\neg Q(y) \vee \neg L(x,y)))$

$(\forall x)(\forall y)\ (\neg P(x) \vee (\neg Q(y) \vee \neg L(x,y)))$

$\neg P(x) \vee \neg Q(z) \vee \neg L(x,z)$

6) $L(a,b)$ 1)和4) $\{y/b\}$
7) $\neg Q(z) \vee \neg L(a,z)$ 3)和5) $\{x/a\}$
8) $\neg L(a,b)$ 2)和7) $\{z/b\}$
9) NIL 6)和8)

归结策略

- **单元优先**：优先对那些包含一个单文字（也称为**单元子句**）的语句进行归结。要试图产生空子句，那么首先完成那些能够产生较短子句的推理。将一个单元语句（如P）和任何其它语句（如$\neg P \vee \neg Q \vee R$）归结时，总是生成一个比其它子句短的子句（在本例中，$\neg Q \vee R$）。

归结策略

- **单元归结**将归结限制为每次归结都必须包含单元子句。单元归结一般来说是不完备的，如果是Horn子句则是完备的。Horn子句的单元归结证明类似于前向链接

归结策略

- OTTER定理证明器（Organized Techniques for Theorem-proving and Effective Research, McCune, 1992）使用了**最佳优先搜索**。它的启发式函数度量了每个子句的“权重”，权重小的优先执行。子句的权重应当与它的规模和难度相关联。单元子句权重轻；所以这种方法可以看作是单元优先策略的一般化。

归结策略

- **支撑集**：坚持归结的每一步都必须涉及至少一个特殊子句集的元素，这个集合被称为*支撑集*。它首先确定一个称为**支撑集**的语句子集。归结式放到支撑集中。如果相对于整个知识库而言支撑集很小，搜索空间将会大幅度缩小。

归结策略

- 对支撑集的错误选择将会使得算法不完备。但是，如果我们选择的支撑集S使得剩余的语句是联合可满足的，那么支撑集的归结是完备的。例如，可用否定查询作为支撑集，假设原始知识库是一致的。（毕竟，如果它不是一致的，则查询事实将是虚无的。）支撑集策略具有生成目标制导的证明树的附加优点，通常易于理解。

归结策略

- **包容**：包容法清除所有被知识库中的已有语句包容（即，比该语句更特例）的语句。例如，如果$P(x)$在KB中，则增加$P(A)$毫无意义，增加$P(A) \vee Q(B)$则更没有意义。包容帮助保持KB的小规模，这样才能帮助保持较小的搜索空间。

前向链接与后向链接

命题逻辑前向链接

- **限定子句**：正好只有一个正文字的子句。它可以是原子语句；或是蕴含语句，蕴含语句的前提是正文字的合取，结论是单个正文字。
- 命题逻辑中，建立在限定子句上的**前向链接算法**：从知识库中的原子语句出发，在前向推理中应用假言推理规则，增加新的原子语句，直到不能进行任何推理。
- 如何应用于一阶限定子句，如何高效地实现。

一阶限定子句

- **一阶限定子句**中，假设变量是全称量化的，书写时省略全称量词
  - King(x) $\wedge$ Greedy(x)$\Rightarrow$ Evil(x)
  - King(John)
  - Greedy(y)
- 有些知识库可写成限定子句的集合，但并非每个知识库都可写为限定子句的集合(单一正文字的限制过于严格)。

一阶限定子句

- 考虑以下问题：
  - 美国法律规定美国人贩卖武器给敌对国家是犯法的。美国的敌对国家Nono有一些导弹，所有这些导弹都是美国人West上校卖给他们的。
- 证明韦斯特是罪犯（criminal）。

一阶限定子句

- 一阶限定子句表示这些事实
- “……美国人贩卖武器给敌对国家是犯法的”
  - $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$ (1)
- “Nono……有导弹。”
  - $\exists x\ Owns(Nono, x) \wedge Missile(x)$
- 消去存在量词被转换成两个限定子句：
  - $Owns(Nono, M_1)$ (2)
  - $Missile(M_1)$ (3)

一阶限定子句

- “所有该国的导弹都购自West上校”
  - $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ (4)
- 还需要知道导弹是武器：
  - $Missile(x) \Rightarrow Weapon(x)$ (5)
- 还需要知道美国的敌人被称为“敌对的”
  - $Enemy(x, America) \Rightarrow Hostile(x)$ (6)
- “West，一个美国人……”
  - $American(West)$ (7)
- “Nono国，美国的敌人……”
  - $Enemy(Nono, America)$ (8)

一阶限定子句

- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$ (1)
- $Owns(Nono, M_1)$ (2)
- $Missile(M_1)$ (3)
- $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ (4)
- $Missile(x) \Rightarrow Weapon(x)$ (5)
- $Enemy(x, America) \Rightarrow Hostile(x)$ (6)
- $American(West)$ (7)
- $Enemy(Nono, America)$ (8)

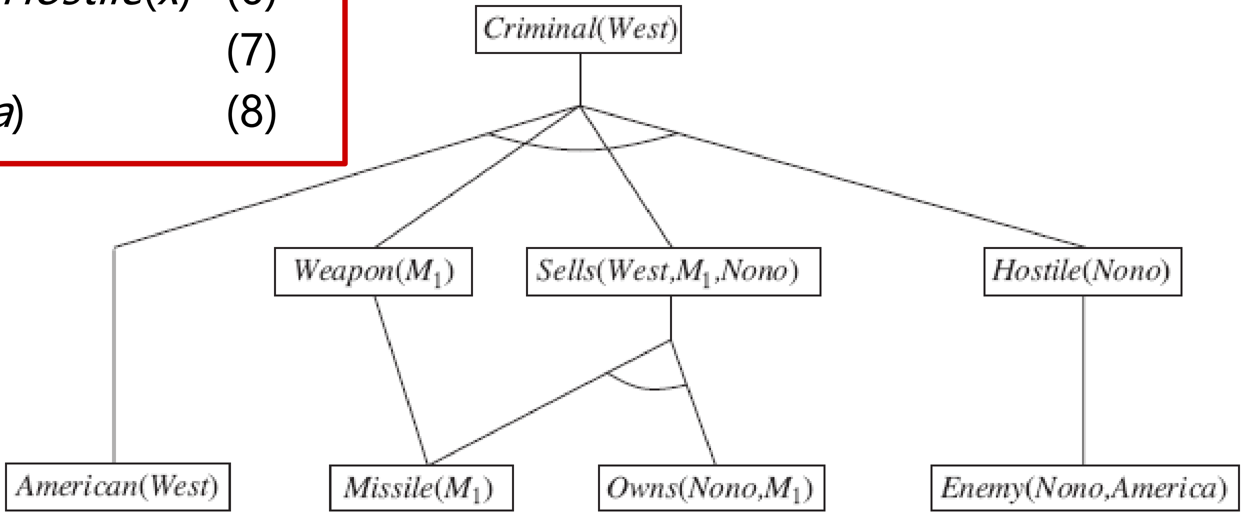
不包含函词，是**数据日志类**知识库。数据日志是一种受限于没有函词的一阶限定语句的语言。没有函词的推理更容易。

数据日志可以表示由关系数据库生成的语句类型。

前向链接算法

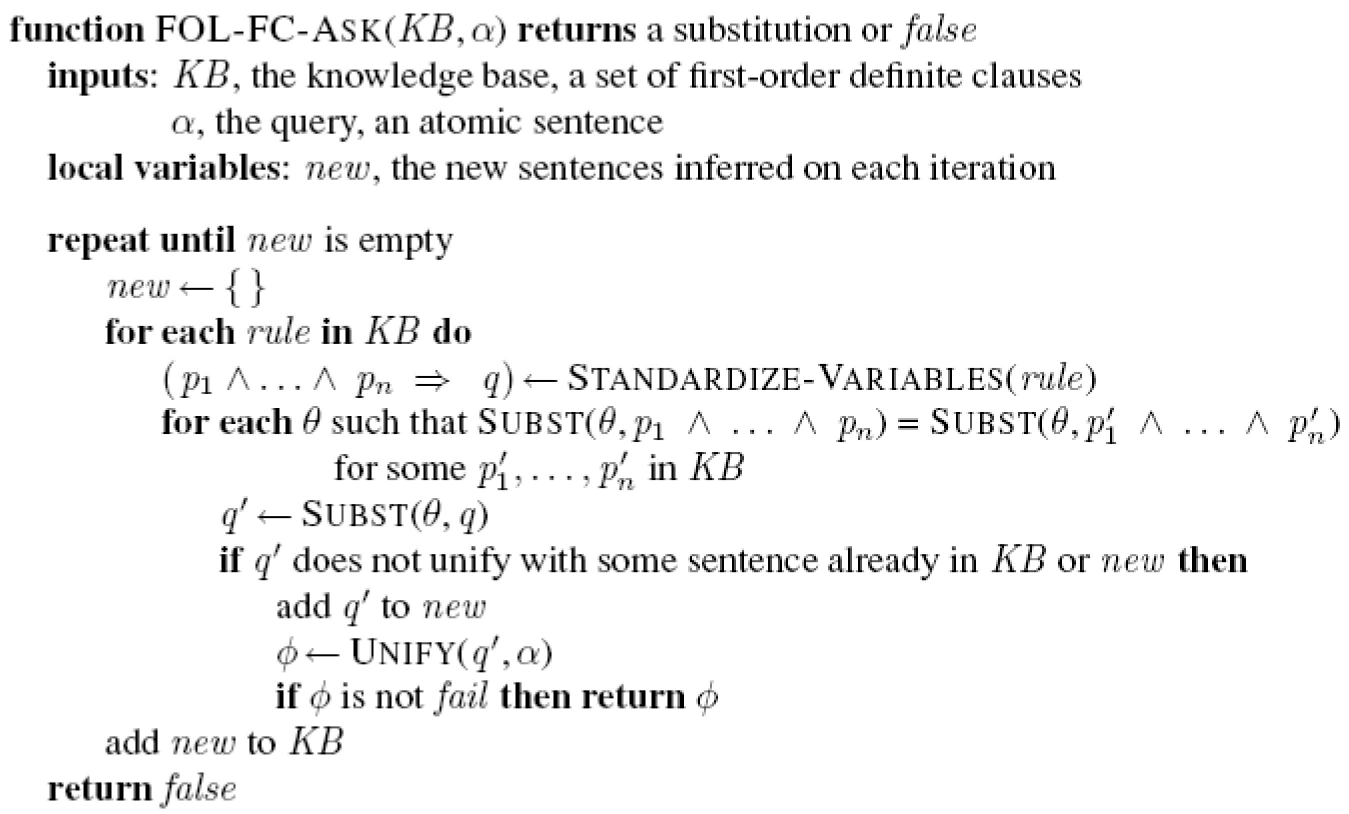
- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$ (1)
- $Owns(Nono, M_1)$ (2)
- $Missile(M_1)$ (3)
- $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ (4)
- $Missile(x) \Rightarrow Weapon(x)$ (5)
- $Enemy(x, America) \Rightarrow Hostile(x)$ (6)
- $American(West)$ (7)
- $Enemy(Nono, America)$ (8)

- 使用前向链接算法进行证明



前向链接算法

□ 前向链接算法FOL-FC-ASK



```
function FOL-FC-ASK($KB, \alpha$) returns a substitution or $false$
  inputs: $KB$, the knowledge base, a set of first-order definite clauses
          $\alpha$, the query, an atomic sentence
  local variables: $new$, the new sentences inferred on each iteration

  repeat until $new$ is empty
      $new \leftarrow \{\}$
      for each $rule$ in $KB$ do
          $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
          for each $\theta$ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
                      for some $p'_1, \ldots, p'_n$ in $KB$
              $q' \leftarrow$ SUBST($\theta, q$)
              if $q'$ does not unify with some sentence already in $KB$ or $new$ then
                  add $q'$ to $new$
                  $\phi \leftarrow$ UNIFY($q', \alpha$)
                  if $\phi$ is not $fail$ then return $\phi$
      add $new$ to $KB$
  return $false$
```

高效的前向链接算法

效率低下的可能原因。

(1)算法每次迭代都要对所有规则重新进行检查，以确定其前提是否已经得到满足，即使每次迭代添加到知识库的内容非常少，也要全部检查。

(2)算法的“内循环”涉及寻找所有可能的合一置换。这一过程称为**模式匹配**，代价昂贵。

(3)算法可能生成很多与目标无关的事实。

```
function FOL-FC-ASK($KB$, $\alpha$) returns a substitution or $false$
  inputs: $KB$, the knowledge base, a set of first-order definite clauses
          $\alpha$, the query, an atomic sentence
  local variables: $new$, the new sentences inferred on each iteration

  repeat until $new$ is empty
      $new \leftarrow \{\}$
      for each $rule$ in $KB$ do
          $(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
          for each $\theta$ such that SUBST($\theta, p_1 \land \ldots \land p_n$) = SUBST($\theta, p'_1 \land \ldots \land p'_n$)
                      for some $p'_1, \ldots, p'_n$ in $KB$
              $q' \leftarrow$ SUBST($\theta, q$)
              if $q'$ does not unify with some sentence already in $KB$ or $new$ then
                  add $q'$ to $new$
                  $\phi \leftarrow$ UNIFY($q', \alpha$)
                  if $\phi$ is not $fail$ then return $\phi$
      add $new$ to $KB$
  return $false$
```

高效的前向链接算法

- 将规则前提与知识库中的已知事实**匹配**

模式匹配

- 例如，考虑规则$Missile(x) \Rightarrow Weapon(x)$
- 需要找出所有能与$Missile(x)$合一的事实；对于已经建立了适当索引的知识库，这一过程对于每个事实都可以在常数时间内完成

高效的前向链接算法

- 考虑规则$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

- **合取排序**问题：对规则前提的合取项进行排序，使总成本最小。

高效的前向链接算法

- 考虑规则$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

- 如果知识库中Nono拥有的对象不多：可以花费常数时间找出Nono拥有的全部对象；然后检查它们是不是导弹。
- 如果知识库中Nono拥有很多的对象，导弹却不多，那么最好是先找出所有的导弹然后检查它们是否为Nono所有。

高效的前向链接算法

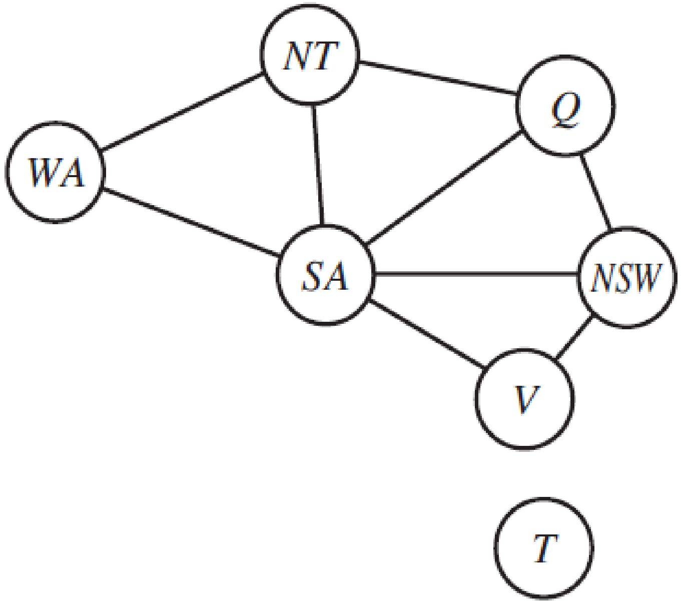
- 寻求最优排序是NP难题，但有好的启发式可用。例如，CSP的最少剩余值（MRV）启发式建议：
- 如果导弹数目少于Nono拥有的对象的数目，那么应对合取项进行排序以便首先搜索导弹
  - $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- 否则，优先搜索Nono拥有的对象
  - $Owns(Nono, x) \wedge Missile(x) \Rightarrow Sells(West, x, Nono)$

高效的前向链接算法

- 模式匹配和约束满足联系紧密。可以将每个合取项看作它所包含的变量上的一个约束，
- 例如$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- 其中$Missile(x)$是x的一元约束。

高效的前向链接算法

- 扩展该思想，可以把有限论域的CSP表示为单个限定子句以及一些相关的基本事实。



(a)

$$\begin{aligned}
&Diff(wa, nt) \wedge Diff(wa, sa) \wedge \\
&\quad Diff(nt, q) \wedge Diff(nt, sa) \wedge \\
&\quad Diff(q, nsw) \wedge Diff(q, sa) \wedge \\
&\quad Diff(nsw, v) \wedge Diff(nsw, sa) \wedge \\
&\quad Diff(v, sa) \Rightarrow Colorable()
\end{aligned}$$

$Diff(Red, Blue)$ $Diff(Red, Green)$
$Diff(Green, Red)$ $Diff(Green, Blue)$
$Diff(Blue, Red)$ $Diff(Blue, Green)$

(b)

显然，只有CSP有解时，才能推导出结论*Colorable*( )。

高效的前向链接算法

- 内循环中的模式匹配是NP难的，但可改善：
- 现实世界中，常见的规则的规模和谓词的参数个数都限于某个常数。需要担心的是数据复杂度：知识库中事实的数量的函数，前向链接具有多项式数据复杂度。
- 每个数据日志子句都可视为在定义一个CSP。如果约束图是一颗树，那么CSP可线性时间求解，模式匹配也是同样的。

高效的前向链接算法

```
function FOL-FC-ASK($KB$, $\alpha$) returns a substitution or $false$
  inputs: $KB$, the knowledge base, a set of first-order definite clauses
          $\alpha$, the query, an atomic sentence
  local variables: $new$, the new sentences inferred on each iteration

  repeat until $new$ is empty
     $new \leftarrow \{\}$
     for each $rule$ in $KB$ do
        $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
        for each $\theta$ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
                    for some $p'_1, \ldots, p'_n$ in $KB$
           $q' \leftarrow$ SUBST($\theta, q$)
           if $q'$ does not unify with some sentence already in $KB$ or $new$ then
              add $q'$ to $new$
              $\phi \leftarrow$ UNIFY($q', \alpha$)
              if $\phi$ is not $fail$ then return $\phi$
     add $new$ to $KB$
  return $false$
```

模式匹配中冗余的规则匹配可消除。

第二次迭代时Missile(x) ⇒ Weapon(x)与Missile(M$_1$)再次匹配，而结论Weapon(M$_1$)已知，所以什么都不会发生

- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$ (1)
- $Owns(Nono, M_1)$ (2)
- $Missile(M_1)$ (3)
- $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ (4)
- $Missile(x) \Rightarrow Weapon(x)$ (5)
- $Enemy(x, America) \Rightarrow Hostile(x)$ (6)
- $American(West)$ (7)
- $Enemy(Nono, America)$ (8)

高效的前向链接算法

- 第t次迭代推理出来的新事实应该由至少一个第t – 1次迭代中推理出来的新事实导出，由此可以避免大量多余的规则匹配。任何不需要来自第t – 1次迭代的新事实的推理，可能在第t – 1次迭代中就已经完成。

增量前向链接

高效的前向链接算法

- 增量前向链接算法。
  - 第t次迭代时，只激活这些规则：前提包含了能与第$t-1$次迭代新推理出的事实p_i`进行合一的合取项p_i。

- 建立合适的索引，快速检索能被已知事实激活的规则

高效的前向链接算法

- 大量的冗余工作：不断重复构造不完全匹配。
- 第一次迭代时$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$和事实$American(West)$之间是不完全匹配。
- 该不完全匹配被舍弃并在后面迭代时重建。
- 好的做法是，保留并逐步完成不完全匹配，而不是舍弃它们。

高效的前向链接算法

无关的事实

- 前向链接允许产生所有基于已知事实的推理，*即使它们与需要达到的目标毫无关系*

高效的前向链接算法

- 利用目标信息重写规则集，从而在前向前推理过程中只考虑相关的变量绑定——**魔法集**。
- 若目标是*Criminal*(*West*)，结论为*Criminal*(*x*)的规则被重写以包含附加的、对*x*的取值进行约束的合取子句：
  - $Magic(x) \land American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
- 事实*Magic*(*West*)也被加入到KB中。这样，即使知识库中包括上百万美国人的事实，在前向推理过程中也只会考虑West。

高效的前向链接算法

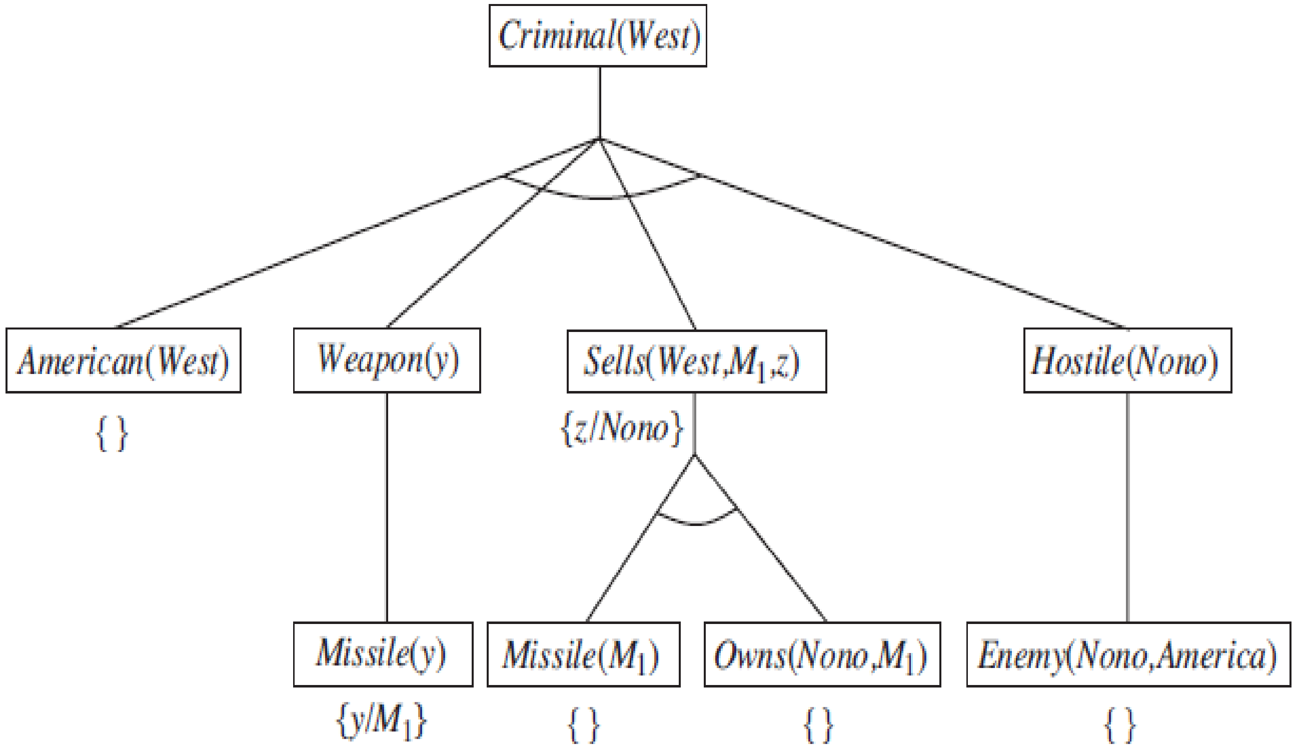
- 避免推导出无关结论的另一个方法是采用反向链接。

后向链接算法

- 后向链接算法
- 用于逻辑程序设计
- 逻辑程序设计与CSP

后向链接算法

- 后向链接是一种深度优先搜索算法
- 一旦合取式中的某个子目标得以成功实现，它的置换要用于后续子目标。



summary

1 代换、全称量词实例化、存在量词实例化
2 合一、CNF、归结推理
3 前向链接与后向链接