推　理

2019年度南京大学“专创融合”特色示范课程培育项目

高 阳

http://cs.nju.edu.cn/rl, 2019.9.17

推 理

从演算到自动推理

高 阳

http://cs.nju.edu.cn/gaoy, 2019.9.17

引言

概念

推理： 从一个或几个已知的判断(前提)逻辑地推论出一个新的判断(结论)的思维形式。这是事物客观联系在意识中的反映。

自动推理： 是主要指机器定理证明，其核心问题是寻找判定公式是否有效的(或者不一致的)通用程序。

推理分单调推理与非单调推理

1930年，Herbrand奠定基本方法；1965年，Robinson建立归结原理。

命题演算

命题演算

命题演算（逻辑学术语）是命题逻辑的公理化，任务是使用演算手段来讨论命题逻辑。

命题演算是一个形式系统，包含(1)由以逻辑运算符结合原子命题来构成代表“命题”的公式，(2)以及允许某些公式建构成“定理”的一套形式“证明规则”。

由Newell和Simon在上世纪60年代，应用到逻辑理论家和通用问题求解器中。

命题演算：语法

命题演算的符号：是命题符号, 命题符号代表命题，是关于现实世界的能分辨真假值的陈述句。

命题符号：P，Q，R，S，T

真值符号：True，False

联结词：$\vee$，$\wedge$，$\neg$，$\rightarrow$，$=$

通过联结词可把多个命题组成合成的命题，也称为合式公式。

命题演算：语义

如两个命题表达式 在任何真值指派下都有相同的值，则称为是等价的

真值表证明：　　$P \rightarrow Q$ 与 $\neg P \vee Q$ 等价。

P	Q	$P \rightarrow Q$
T	T	T
F	T	T
T	F	F
F	F	T

对于命题表达式 P，Q，R

$\neg(\neg P) = P$；　$(P \vee Q) = (\neg P \rightarrow Q)$

命题演算：推理

逻辑理论家

使用统一的表示法和可靠的推理规则，并应用了数个策略或启发式方法来指导求解过程。

推理规则：代换、置换、分离。

推理策略：以宽度优先、目标驱动的方式将这些推理规则应用到要证明的定理中，用于尝试找到一系列操作，最终可以推导到公理或已知为真的定理。

命题演算：例子

逻辑理论家

$$(p \rightarrow \neg p) \rightarrow \neg p$$

1. $(A \vee A) \rightarrow A$　　匹配过程确定5个公理中最“好”的

2. $(\neg A \vee \neg A) \rightarrow \neg A$　　用 $\neg A$ 代换 A

3. $(A \rightarrow \neg A) \rightarrow \neg A$　　用$\rightarrow$替换$\vee$ 和$\neg$

4. $(p \rightarrow \neg p) \rightarrow \neg p$　　用p 代换 A.

证毕

子句和子句形

文字是原子命题或其否定式

子句是文字的析取

合取范式(CNF) $(L_{1_1} \vee \ldots \vee L_{1n_1}) \wedge \ldots \wedge (L_{m_1} \vee \ldots \vee L_{mn_m})$

析取范式(DNF) $(L_{1_1} \wedge \ldots \wedge L_{1n_1}) \vee \ldots \vee (L_{m_1} \wedge \ldots \wedge L_{mn_m})$

定理：对任意公式，都有与之等值的合取范式和析取范式

转换DNF方法：一般方法或者真值表方法

子句范式化

双重否定法

德.摩根定律

$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

$$\neg(P \vee Q) = \neg P \wedge \neg Q$$

谓词演算

谓词演算

命题逻辑的缺陷

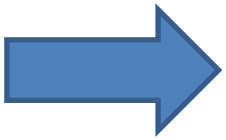
例一：王平是大学生

可以分解为主语(王平)和谓语(是大学生)，

命题逻辑反映不出这一特点。

例二：王平是大学生和李明是大学生

大学生这一共性在命题逻辑中也表示不出来。



谓词演算

谓词演算：语法

字母表

（1）英文字母组合，包括大写与小写

（2）数字集合0, 1, ⋯, 9

（3）下划线

项

每一个变量和常数都是项。如果u0, u1, · · · un是项，则f(u0, u1, , un)也是。

原子公式

如果u0, u1, · · · un是项，则R(u0, u1, un)是原子公式。

公式

每一个原子公式都是公式； 如果φ和ψ是公式，则 ¬φ, (φ)， ∀xφ, ∃xφ, φ∨ψ, φ∧ψ 也是。

如果ψ是一个公式并且是φ中的一串连续的符号， 则ψ是φ的子公式。

谓词演算：语义

表达式的真值依赖于常元、变元、谓词、函词到论域中的映射；在论域中的关系的真假决定了相应表达式的真假。

例如：

friends(george, susie)

friends(george, kate)

谓词演算：语义

一个论域D上的解释：

假设论域D是一个非空集合，在D上的一个解释把论域D的实体指派给一个谓词演算表达式的每一个常元、变元、谓词及函词符号，于是有：

1）每一个常元指派了D的一个元素。

2）对每一个变元，指派D的一个非空集合，这是该变元的变域。

3）每个n元谓词P定义在论域D中的n个参数上，并定义了从D^n到{T，F}的一个映射。

4) 每个m元函词f定义在论域D的m个参数上，并定义了从D^m到D的一个映射。

在一种解释下，一个表达式的意义是在该解释下的一个真值指派。

谓词演算：语义

一个论域D上的解释：

常元	D的某个元素
变元	D的某个非空集合
n元谓词P	D^n到{T，F}的一个映射
m元函词f	D^m到D的一个映射

谓词演算：语义

否定与全称量词、存在量词之间的关系

对于谓词 P, Q, 变元 x, y有：

- $\neg\exists x\, P(x) = \forall x \neg P(x)$
- $\neg\forall x P(x) = \exists x \neg P(x)$
- $\exists x P(x) = \exists y P(y)$
- $\forall x Q(x) = \forall y Q(y)$
- $\forall x(P(x) \wedge Q(x)) = \forall x P(x) \wedge \forall y Q(y)$
- $\exists x(P(x) \vee Q(x)) = \exists x P(x) \vee \exists y Q(y)$

谓词演算：推理规则

- 从其他谓词演算命题产生新的谓词演算命题的机械方法
  - 产生基于给定逻辑断言的句法形式的新命题
  - 当每个由逻辑表达式集S上的推理规则产生的命题X都是S的逻辑结果, 则称该逻辑规则是合理的
- 五大推理规则

取式假言推理、拒式假言推理、与消除、与引入、全称例化

谓词演算：推理规则

- 取式假言推理： 如果已知语句P和P→Q为真，那么Q为真
- 拒式假言推理： 如果已知P→Q为真，且Q为假，那么P为假
- 与消除： P∧Q为真，那么P、Q分别为真
- 与引入：P和Q都为真， P∧Q为真
- 全称例化： ∀ X p(X)为真，且a为X定义域中一值，p(a)为真

P	Q	P → Q
T	T	T
F	T	T
T	F	F
F	F	T

谓词演算：推理规则

假言推理：如果命题P，P → Q为真，应用假言推理得出Q为真。

S: $\forall x\ human(x) \rightarrow mortal(x)$.

$human(Socrates)$.

x: Socrates.

$human(Socrates) \rightarrow mortal(Socrates)$

? $human(x)$ ⟸ Socrates

合一算法

合一

合一

- 合一：判断什么样的替换可以使两个谓词表达式匹配的算法
  - 合一表明了两个或多个表达式在什么条件下可以称为等价的。
- 替换：

一个替换(Substitution)就是形如{t1/x1,t2/x2,….tn/xn}的有限集合，x1,x2.,,,xn是互不相同的个体变元，ti不同于xi, xi也不循环出现在tj中

如：{a/x,g(y)/y,f(g(b))/z}　正确！

{g(y)/x,f(x)/y}　错误！

合一

斯柯伦标准化：去掉所有的存在量词

$\exists Z(foo(Y, Z))$ $\quad\quad$ $foo(Y, k)$

$\exists X(dog(X))$ $\quad\quad$ $dog(fido)$

斯柯伦常元

如果谓词中含有多个参数，而∃约束变元在∀约束变元的约束范围内，则∃约束变元必须是那些其他变元的函数

$(\forall X)(\exists Y)(mother(X,Y))$

$(\forall X)\ (mother(X,m(X))$

合一

- 组合(composition)：

在合一过程中，如果先后产生S和S’的替换，那么将S中的某个元素应用S’，所产生的新的替换，称为组合。

如：{X/Y, W/Z}, {V/X}, {a/V, f(b)/W}

组合过程：{X/Y, W/Z}, {V/X}组合产生{V/Y, W/Z}

{V/Y, W/Z},{a/V, f(b)/W} 组合产生{a/Y, f(b)/Z}

最一般的合一式

- 合一式的泛化：

观察：man(X), man(Y)

不唯一的合一结果： {Mike/X, Mike/Y}，其中Mike是常元

{Male/X, Male/Y}，其中Male是变元

- 最一般的合一式(MGU, MOST GENERALIZE UNIFY)：

如果g是表达式E的最一般合一式，那么对于任何的其他合一式s，都存在另一个合一式s’，使Es=Egs’。其中Es和Egs’是应用到表达式E的合一式的组合。

合一算法

算法

1. 递归地对表达式的初始成分合一。如果成功，这次合一返回的任何代入式被用到两个表达式的剩下部分，然后以这两个表达式为参数。

2. 当两个参数之一为一个符号(谓词名，函词名，变元，常元)，或两个表达式的每一元素都已匹配了，算法终止。

合一算法

算法 UNIFY

case

E1, E2 或者是常元或者是空表:　%递归终止

　　If E1 ＝ E2 then return { };

　　else return FAIL;

E1是一个变元:

　　if E1在E2中出现 then return FAIL;

　　else return {E2/E1};

E2是一个变元:

　　if E2在E1中出现 then return FAIL;

　　else return {E1/E2};

其他情况: % E1和E2都是表

合一算法

算法

```
begin
    HE1:= E1的第一个元素;
    HE2:= E2的第一个元素;
    SUBS1:= unify (HE1, HE2);
    if (SUBS1 ＝ FAIL) then return FAIL;
    TE1:= apply (SUBS1, E1的后半部);
    TE2:= apply (SUBS1, E2的后半部);
    SUBS2:= unify (TE1, TE2 );
    if  (SUBS2 ＝ FAIL) then  return FAIL;
    else return SUBS1与SUBS2 的并集;
end
```

例子

- parents(X, father(X), mother(bill))
- parents(bill, father(bill), Y)

- {bill/X}
- parents(bill, father(bill), mother(bill))
- parents(bill, father(bill), Y)

- {mother(bill)/Y}
- parents(bill, father(bill), mother(bill))
- parents(bill, father(bill), mother(bill))

- {bill/X, mother(bill)/Y}　组合

```
begin
    HE1:= E1的第一个元素;
    HE2:= E2的第一个元素;
    SUBS1:= unify (HE1, HE2);
    if (SUBS1 ＝ FAIL) then return FAIL;
    TE1:= apply (SUBS1, E1的后半部);
    TE2:= apply (SUBS1, E2的后半部);
    SUBS2:= unify (TE1, TE2 );
    if  (SUBS2 ＝ FAIL) then  return FAIL;
    else return SUBS1与SUBS2 的并集;
end
```

归结

归结法的基本思想

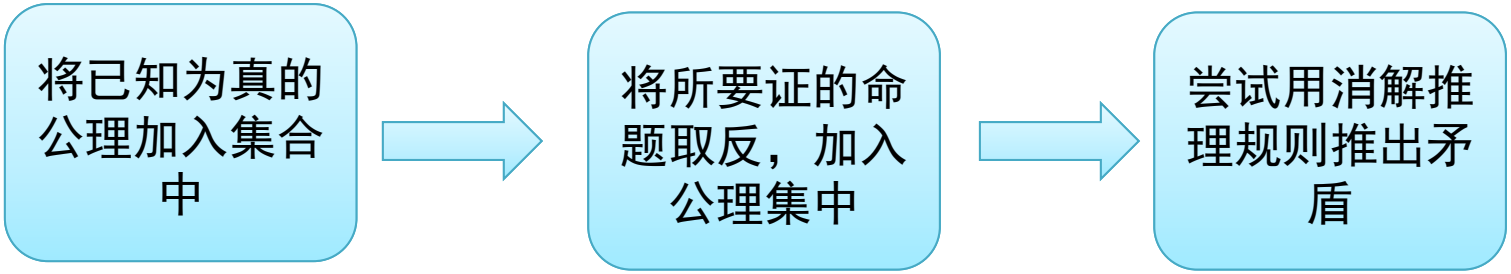
- **反证法：**为了证明一个命题P恒真，则证明其反命题¬P恒假。即不存在使得¬P为真的解释。

- 设$F_1,F_2,\ldots F_n,G$为公式，G为$F_1,\ldots,F_n$的逻辑推论，当且仅当公式$(F_1 \wedge \ldots \wedge F_n \wedge \neg G)$是不可满足的。

归结(消解)

- 消解是一种应用于谓词演算中的定理证明技术，是人工智能问题求解的一个组成部分。消解描述了如何用最少的合一次数在一个子句数据库中发现矛盾的方法。

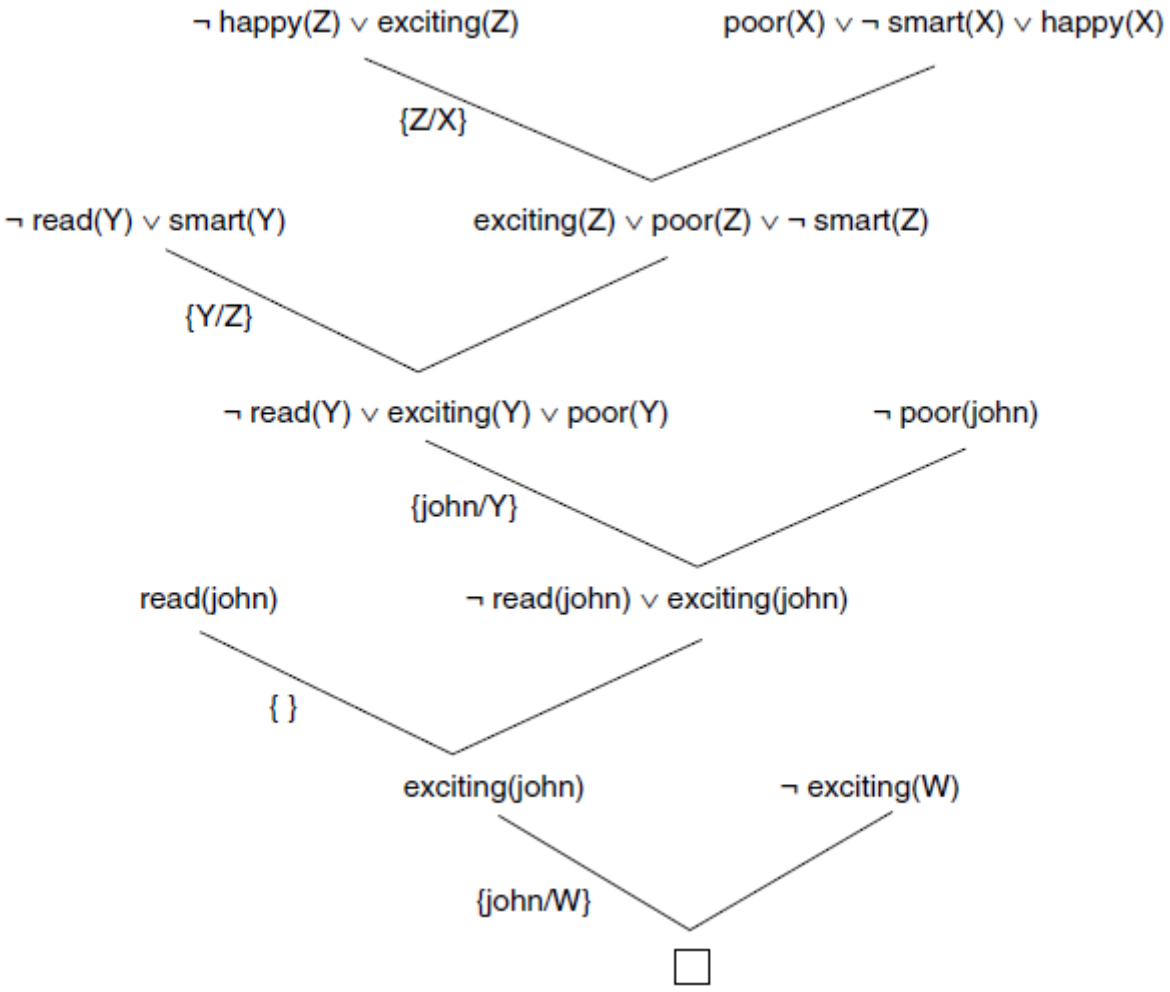
- 具体方法为(**目标驱动**)



稍复杂的例子：幸福生活

- 幸福生活
  - 所有不贫穷而且聪明的人是快乐的。读书的人不愚蠢。约翰能读书且不贫穷。快乐的人过着幸福的生活。能找到过着幸福生活的人吗？
- 谓词描述
  - $\forall X(\neg poor(X) \wedge smart(X) \rightarrow happy(X))$
  - $\forall Y(read(Y) \rightarrow smart(Y))$
  - $read(john) \wedge \neg poor(john)$
  - $\forall Z(happy(Z) \rightarrow exciting(Z))$
  - $\neg \exists W(exciting(W))$

归结(消解)


¬ happy(Z) ∨ exciting(Z)
poor(X) ∨ ¬ smart(X) ∨ happy(X)
{Z/X}
¬ read(Y) ∨ smart(Y)
exciting(Z) ∨ poor(Z) ∨ ¬ smart(Z)
{Y/Z}
¬ read(Y) ∨ exciting(Y) ∨ poor(Y)
¬ poor(john)
{john/Y}
read(john)
¬ read(john) ∨ exciting(john)
{ }
exciting(john)
¬ exciting(W)
{john/W}
□

归结(消解)

消解否证包含以下步骤：

- 具体方法为把前提或公理转换成子句形式。
- 把求证目标的否定的子句形式加到公理集合中。
- 对所有这些子句进行消解，产生它们的逻辑结果子句。
- 用产生空子句的方法来得出矛盾。

子句形-SKOLEM标准形

SKOLEM标准型：

- 没有存在量词的公式。

- 设G是一阶逻辑中的公式，将其转化为SKOLEM标准型，其给出的子句集S为公式G的子句集。

转换子句

$(\forall X)([a(X) \wedge b(X)] \rightarrow [c(X,I) \wedge (\exists Y)((\exists Z)[c(Y,Z)] \rightarrow d(X,Y))]) \vee (\forall X)(e(X))$

- 消去蕴含： $a \rightarrow b \equiv \neg a \vee b$

$(\forall X)(\neg[a(X) \wedge b(X)] \vee [c(X,l) \wedge (\exists Y)((\exists Z)[\neg\, c(Y,Z)] \vee d(X,Y))] \vee (\forall X)(e(X))$

- 将否定式降至原子式：

$\neg(\neg\, a) \equiv a$　　$\neg(\exists X)\, a(X) \equiv (\forall X) \neg a(X)$　　$\neg(\forall X)\, b(X) \equiv (\exists X) \neg b(X)$

$\neg(a \wedge b) \equiv \neg a \vee \neg b$　　$\neg(a \vee b) \equiv \neg a \wedge \neg\, b$

应用得到

$(\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge (\exists Y)((\exists Z)[\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall X)(e(X))$

转换子句

$(\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge (\exists Y)((\exists Z)[\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall X)(e(X))$

- 重命名所有变量标准化：

$((\forall X)a(X) \vee (\forall X)b(X)) \equiv (\forall X)a(X) \vee (\forall Y)b(Y)$

应用得到

$(\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge (\exists Y)((\exists Z)[\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall W)(e(W))$

- 将所有量词左移：

$(\forall X)(\exists Y)(\exists Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge [\neg c(Y,Z)] \vee d(X,Y)]) \vee (e(W))$

转换子句

$(\forall X)(\exists Y)(\exists Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge [\neg c(Y,Z)] \vee d(X,Y)]) \vee (e(W))$

- 消除存在量词：

$(\exists X)(dog(X))$ 可以置换为dog(fido)

如果存在量词在全称量词作用范围内，则存在量化变量为其他变量的函数

$(\forall X)\ (\exists Y)\ (mother(X,Y))$

$(\forall X)\ mother(X, m(X))$

另一个例子

$(\forall X)(\forall Y)(\exists Z)(\forall W)(foo(X,Y,Z,W))$

$(\forall X)(\forall Y)(\forall W)(foo(X,Y,f(X,Y),W))$.

应用到原例

$(\forall X)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge [\neg c(f(X),g(X))] \vee d(X,f(X)))]) \vee (e(W))$

转换子句

$(\forall X)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge [\neg c(f(X),g(X))] \vee d(X,f(X)))]) \vee (e(W))$

- 去掉所有全称量词：

$([\neg a(X) \vee \neg b(X)] \vee [c(X,l) \wedge [\neg c(f(X),g(X))] \vee d(X,f(X)))]) \vee (e(W))$

- 将表达式转化为析取子句的合取形式：

$a \vee (b \vee c) = (a \vee b) \vee c$
$a \wedge (b \wedge c) = (a \wedge b) \wedge c$
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

应用到原例

$[\neg a(X) \vee \neg b(X) \vee c(X,l) \vee e(W)] \wedge$
$[\neg a(X) \vee \neg b(X) \vee \neg c(f(X),g(X)) \vee d(X,f(X)) \vee e(W)]$

转换子句

- 将合取分为单独的子句：
  - $\neg a(X) \vee \neg b(X) \vee c(X,l) \vee e(W)$
  - $\neg a(X) \vee \neg b(X) \vee \neg c(f(X),g(X)) \vee d(X,f(X)) \vee e(W)$
- 再次标准化变量：

$$(\forall X)\,(a(X) \wedge b(X)) \equiv (\forall X)\,a\,(X) \wedge (\forall Y)\,b(Y)$$

应用到原例

- $\neg a(X) \vee \neg b(X) \vee c(X,l) \vee e(W)$
- $\neg a(U) \vee \neg b(U) \vee \neg c(f(U),g(U)) \vee d(X,f(U)) \vee e(V)$

简单的例子：死狗问题

谓词演算公式

- All dogs are animals

  $\forall (X)\ (dog(X) \rightarrow animal(X))$

- Fido is a dog

  dog (fido)

- All animals will die

  $\forall (Y)\ (animal(Y) \rightarrow die(Y))$

- 证：Fido will die

  die (fido)

子句形式

- All dogs are animals

  $\neg dog(X) \vee animal(X))$

- Fido is a dog

  dog (fido)

- All animals will die

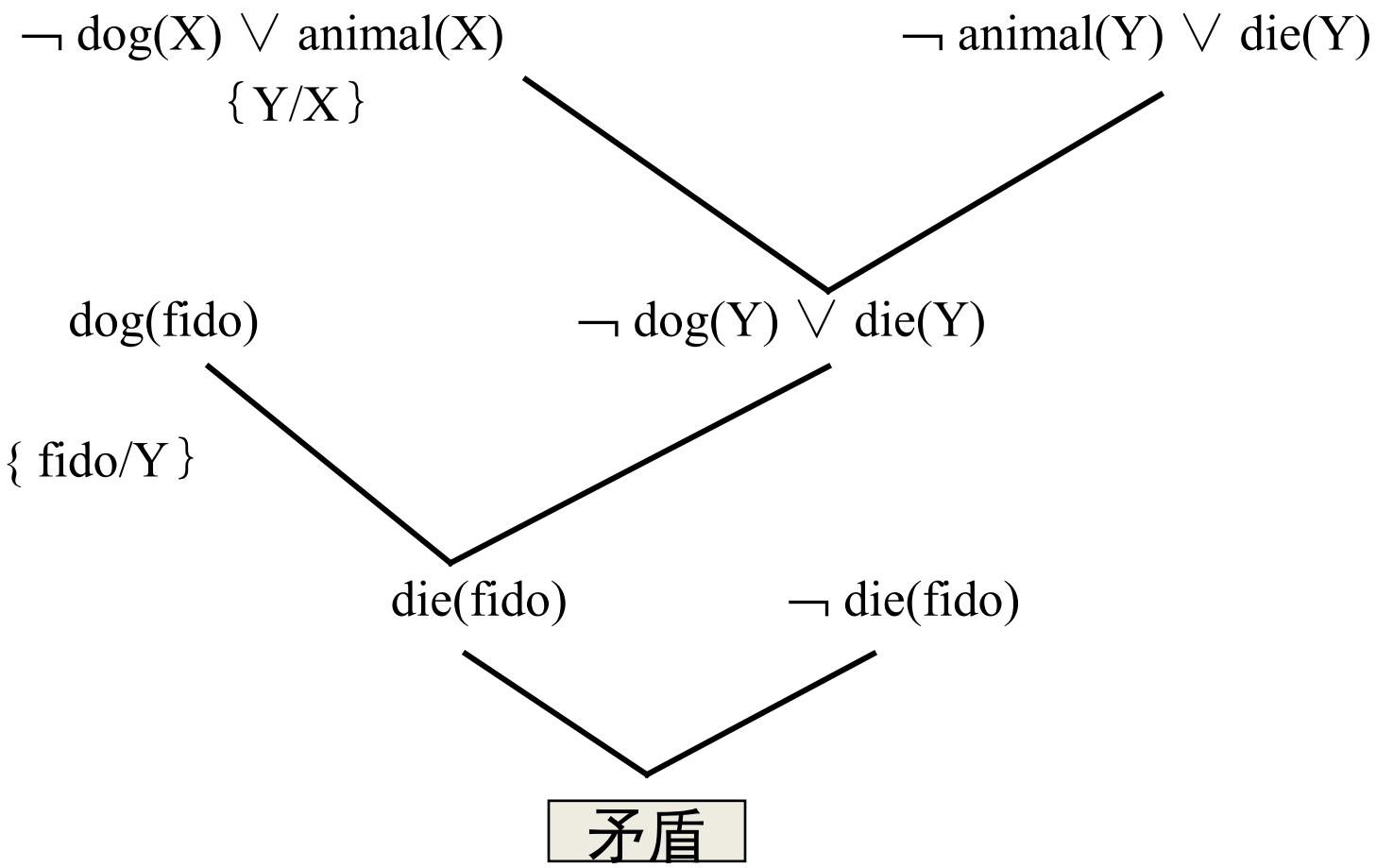
  $\neg\ animal(Y) \vee die(Y))$

- 证：Fido will die

  die (fido)

对目标取反

简单的例子：死狗问题



“死狗”问题消解证明

稍复杂的例子：幸福生活

- 幸福生活
  - 所有不贫穷而且聪明的人是快乐的。读书的人不愚蠢。约翰能读书且不贫穷。快乐的人过着幸福的生活。能找到过着幸福生活的人吗？
- 谓词描述
  - $\forall X(\neg poor(X) \wedge smart(X)) \rightarrow happy(X)$
  - $\forall Y(read(Y) \rightarrow smart(Y))$
  - $read(john) \wedge \neg poor(john)$
  - $\forall Z(happy(Z) \rightarrow exciting(Z))$
  - $\neg \exists W(exciting(W))$

稍复杂的例子：幸福生活

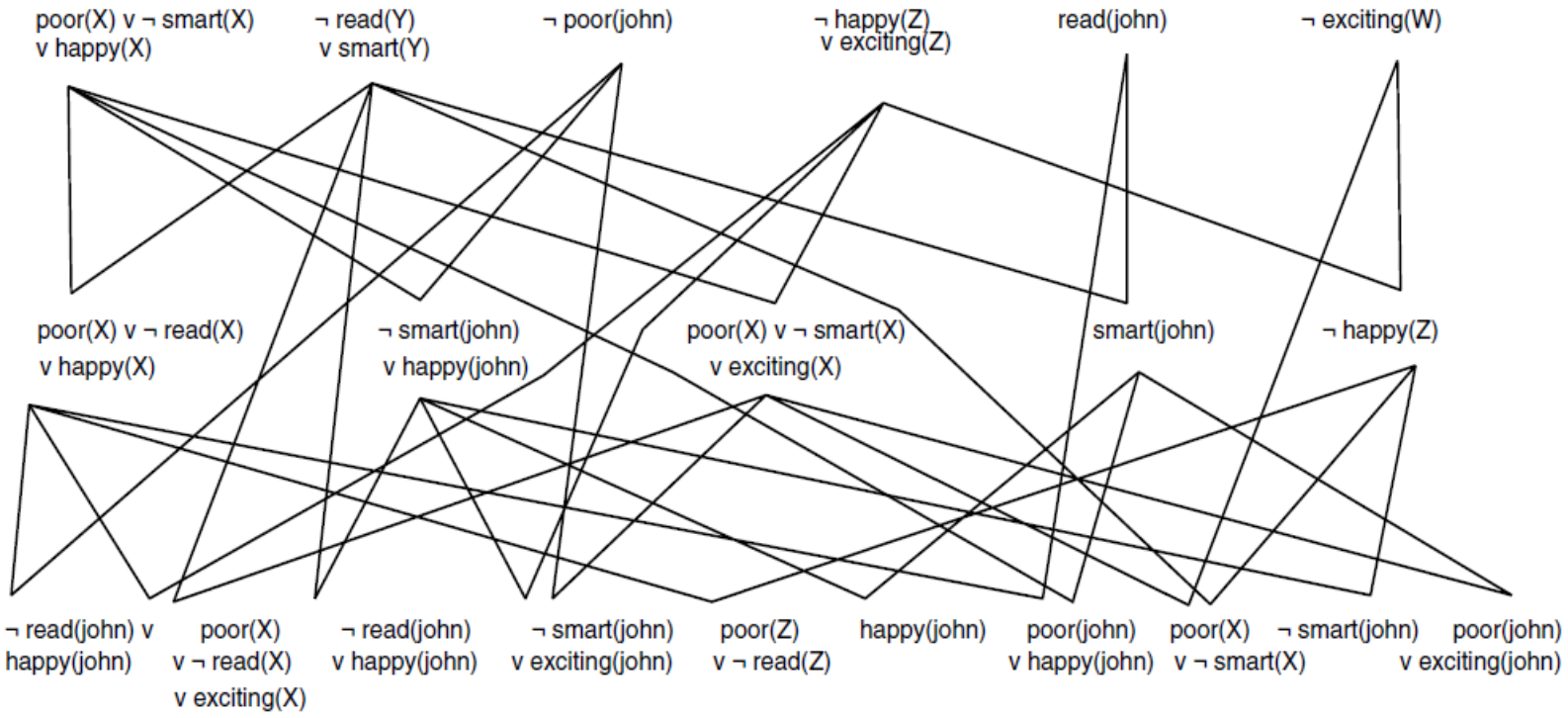
- 子句转换
  - ∀X¬ (¬poor(X) ∧ smart(X)) ∨ happy(X))
    - poor(X) ∨ ¬smart(X) ∨ happy(X)
  - ∀Y(read(Y)→smart(Y))
    - ¬read(Y) ∨ smart(Y))
  - read(john)∧ ¬poor(john)
    - read(john) ¬ poor(john)
  - ∀Z(happy(Z)→exciting(Z))
    - ¬ happy(Z) ∨ exciting(Z)
  - ¬∃W(exciting(W))
    - ¬exciting(W)

归结(消解)的策略

本质和搜索策略相同：

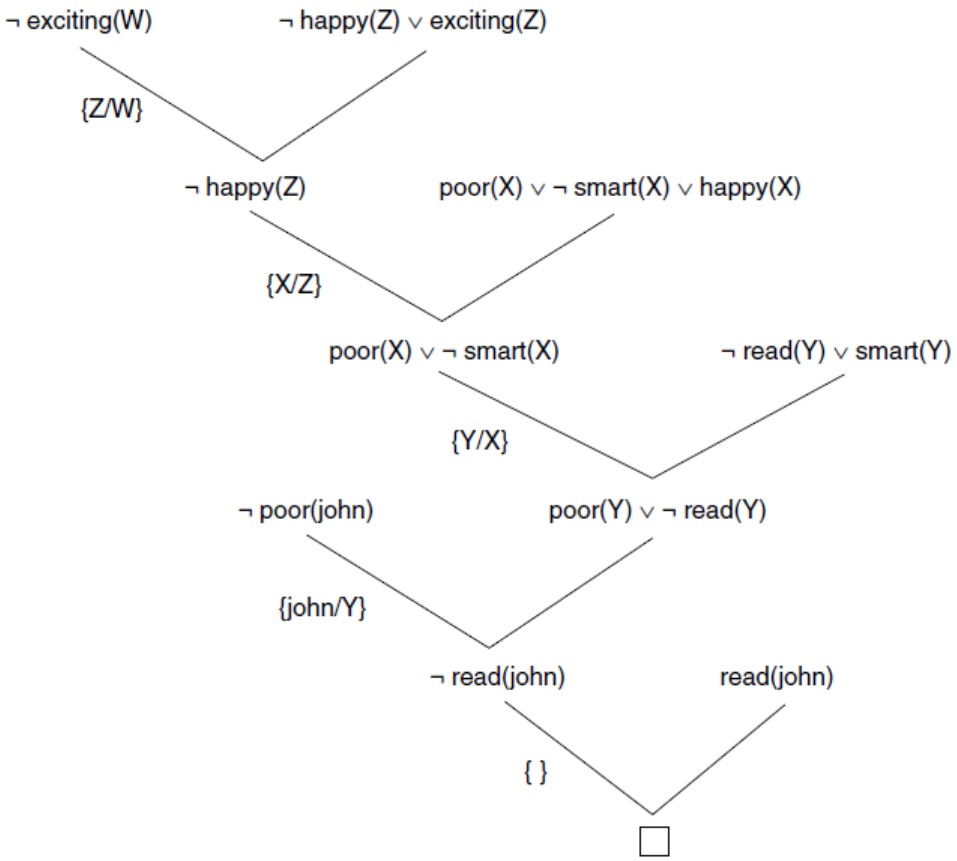
- 宽度优先策略：见下页
- 支持策略：见下页
- 单子句优先策略：见下页
- 线性输入：参与归结的子句至少有一个是初始子句集中的子句；该策略不完备。
- 祖先过滤：参与归结的子句至少有一个是初始子句集中的子句，或者是另一个子句的祖先；该策略完备。

归结(消解)的宽度优先策略



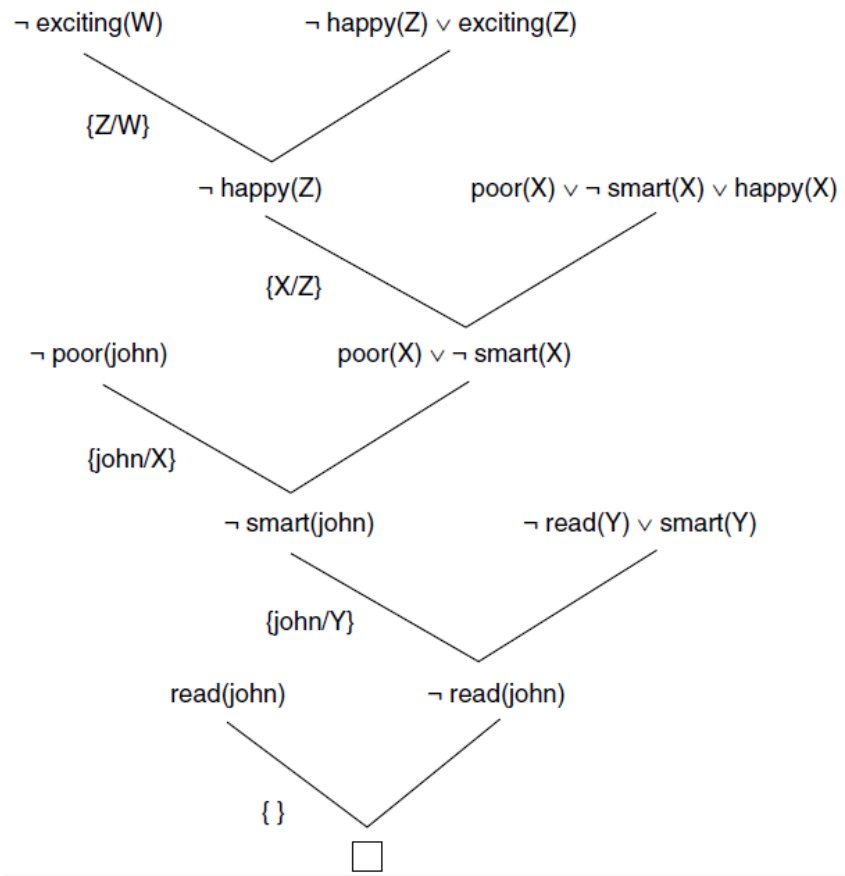
宽度优先：时空开销大，保证能发现最短的解路径

归结(消解)的支持策略



- 支持策略
  - 本策略要求每次归结的归结式之一至少有一个是由目标公式的否定所得到的子句，或者是它们的后裔。
  - 可以证明，该策略是完备的。

归结(消解)的单子句优先策略



- 单个优先策略
  - 只要存在个体子句(一个文字的子句)就是用个体子句归结。
  - 单个优先策略与成组支持策略一起使用，可以得到了更加有效的完备策略。

作业

1. 某村农民王某被害，有四个嫌疑犯A，B，C，D，公安局派出五个侦察员，他们带回的信息各不一样。甲说A，B中至少有一人作案；乙说B，C中至少有一人作案；丙说C，D中至少有一人作案；丁说A，C中至少有一人与此案无关；戊说B，D中至少有一人与此案无关。如果这五个侦察员的话都是可靠的，试用消解法求出谁是罪犯。（请分别用命题演算和谓词演算实现，并理解两种方法之间的差异）

总结

1. 命题演算
2. 谓词演算
3. 合一及最一般的合一式
4. 归结(消解)以及消解策略
5. 与或图

思考和讨论

1. 合一算法中的组合只支持结合律，不支持交换律？
2. 请自行举例，理解最一般的合一式。
3. 在消解策略中，有一种叫做“线性输入策略”。请使用反例说明它是不完备的。
4. 请自学HORN子句。
5. 请自学和了解LISP和PROLOG的发展历史，和基本使用用法。
6. 了解几何自动图里。

实验

1. 实现合一算法和归结消解算法。递归实现合一算法，并分析不同消解策略的效率。采用所实现算法，分析以下问题的结果。

假设：任何拿到学位并找到工作的人都是幸福的，任何认真学习或者家庭背景良好的人都可以拿到学位，李一天不认真学习但家庭背景良好，任何家庭背景良好的人都能找到工作。

求证：李一天是幸福的。

要求：可以用任何语言，需要有日志说明合一中间结果和最终结果，消解的搜索路径，实验报告分析不同消解策略的效率。10月22日前发给助教，检查。

作业提交到公共邮箱: nju_course_ai_2019@163.com

邮件主题格式：姓名_学号_第X次实验

谢 谢 ！