提纲

- **一阶逻辑中的推断**
  - 命题推断与一阶推断
  - 合一与一阶推断
  - 前向链接与反向链接
  - 归结
- **知识表示**
  - 本体论工程
- **自动规划**
  - 经典规划
  - 分层规划

全称量词实例化（universal instantiation，UI）：通过用基本项（没有变量的项）置换全称量化的变量来推断任意语句。

我们使用**置换**来形式化地写出推断规则。令 $\text{SUBST}(\theta, \alpha)$ 表示对语句 α 应用置换 θ 后的语句，则对于任意变量 v 和基本项 g，规则写作：

$$\frac{\forall v\ \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

例如，$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ 实例化可得

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$
$$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$$

存在量词实例化（existential instantiation）用一个新的**常量符号**替换存在量化的变量。其形式化描述如下：对于任意语句 α、变量 v 和未在知识库其他地方出现的常量符号 k:

$$\frac{\exists v\ \alpha}{\text{SUBST}(\{v/k\},\alpha)}$$

例如，$\exists x\ Crown(x) \wedge OnHead(x, John)$ 实例化可得

$$Crown(C_1) \wedge OnHead(C_1, John)$$

只要 C_1 未在知识库的其他地方出现。

存在语句表明存在满足某个条件的对象，运用存在实例化就是给这个对象命名。这个名称不能已经属于其他对象。在逻辑中，这个新的名称被称为**斯科伦常量（Skolem constant）**。

约简为命题推断

假设我们的知识库仅含有语句：

$$\forall x \quad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

且对象仅有 John 和 Richard。用所有可能的置换，对第一条语句应用全称量词实例化：

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

新的知识库是**命题化**的，命题符号为：

$$King(John),\ Greedy(John),\ Evil(John),\ King(Richard)$$ 等

约简为命题推断

假设我们的知识库仅含有语句：

$$\forall x \quad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

且对象仅有 John 和 Richard。用所有可能的置换，对第一条语句应用全称量词实例化：

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

新的知识库是**命题化**的，命题符号为：

$$King(John),\ Greedy(John),\ Evil(John),\ King(Richard)$$ 等

一阶逻辑的蕴含问题是**半可判定的**，也就是，存在能判定所有蕴含的语句的算法，却不存在能够判定所有不蕴含的语句的算法。

合一与一阶推断

一般化肯定前件（generalized Modus Ponens）：对于原子语句p_i、p_i'和 q，存在置换θ使得对所有i有 $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$，则

$$\frac{p_1', p_2', \cdots, p_n', (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

这条规则有n+1个前提：n个原子语句 p_i' 和一个蕴涵式。结论是对后件 q 运用置换 θ的结果。例如，

p_1' is $King(John)$　　p_1 is $King(x)$
p_2' is $Greedy(y)$　　p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$　　q is $Evil(x)$
$q\theta$ is $Evil(John)$

合一与一阶推断

南京大学 NANJING UNIVERSITY

一般化肯定前件的可靠性

对于任意语句 p（假设其变量是全称量化的）和任意置换 θ:

$$p \models \text{SUBST}(\theta, p)$$

因此可以从 p_1' ... p_n' 推得:

$$\text{SUBST}(\theta, p_1') \wedge \cdots \wedge \text{SUBST}(\theta, p_n')$$

从蕴涵式 $p_1 \wedge \cdots \wedge p_n \Rightarrow q$ 可以推得:

$$\text{SUBST}(\theta, p_1) \wedge \cdots \wedge \text{SUBST}(\theta, p_n) \Rightarrow \text{SUBST}(\theta, q)$$

最后根据 $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$ 和肯定前件可得: $\text{SUBST}(\theta, q)$

合一

在一般化肯定前件中我们能够立即获得推断，如果我们能够找到一个置换 θ 使得 $King(x)$ 和 $Greedy(x)$ 匹配 $King(John)$ 和 $Greedy(y)$

这里 $\theta = \{x/John, y/John\}$

这一过程被称作**合并（unification）**。合一算法Unify接收两条语句作为输入，如果存在置换，则为它们返回一个合一子（unifier）（即这个置换）：

$$\text{UNIFY}(p,q)=\theta \text{ 其中 } \text{SUBST}(\theta,p)=\text{SUBST}(\theta,q)$$

p	q	θ
$Knows(John,x)$	$Knows(John,Jane)$	$\{x/Jane\}$
$Knows(John,x)$	$Knows(y,OJ)$	$\{x/OJ, y/John\}$
$Knows(John,x)$	$Knows(y,Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John,x)$	$Knows(x,OJ)$	$fail$

标准化分离（Standardizing apart）消除变量重名，例如，$Knows(z_{17}, OJ)$

合一与一阶推断

南京大学 NANJING UNIVERSITY

合一算法

function UNIFY(x, y, θ=*empty*) **returns** 使x和y相同的置换，或*failure*

  if θ = *failure* **then return** *failure*

  else if x = y **then return** θ

  else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)

  else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)

  else if COMPOUND?(x) **and** COMPOUND?(y) **then**

    return UNIFY(ARGS(x), ARGS(y), UNIFY(OP(x), OP(y), θ))

  else if LIST?(x) **and** LIST?(y) **then**

    return UNIFY(REST(x), REST(y), UNIFY(FIRST(x), FIRST(y), θ))

  else return *failure*

function UNIFY-VAR(*var*, x, θ) **returns** 一个置换

  if 对于一些*val*有 {*var*/*val*} $\in\theta$ **then return** UNIFY(*val*, x, θ)

  else if 对于一些*val*有 {x/*val*} $\in\theta$ **then return** UNIFY(*var*, *val*, θ)

  else if OCCUR-CHECK?(*var*, x) **then return** *failure*

  else return 将{*var*/x}添加到θ

前向链接

南京大学 NANJING UNIVERSITY

一阶确定子句

一阶确定子句是文字的析取式，其中**必须有且仅有一个正文字**。这意味着确定子句要么是原子的，要么是前件为正文字的合取、后件为单个正文字的蕴涵式。存在量词在此处不能使用，而全称量词则被隐式地表示。典型的一阶逻辑确定子句如下：

$$King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

我们用一阶确定子句表示如下问题：

法律规定，美国人将武器出售给敌对国家是犯罪行为。诺诺（Nono）国是美国的敌人，它拥有一些导弹，所有导弹都是韦斯特（West）上校出售给它的，而韦斯特上校是美国人。

证明韦斯特上校有罪。

前向链接

南京大学 NANJING UNIVERSITY

一阶确定子句

我们用一阶确定子句表示这些事实：

……美国人将武器出售给敌对国家是犯罪行为：

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

诺诺国……拥有一些导弹，即，$\exists x\ Owns(Nono, x) \wedge Missile(x)$：

$Owns(Nono, M_1)$ 和 $Missile(M_1)$

……所有导弹都是韦斯特上校出售给它的

$\forall x\ \ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

导弹是武器：

$Missile(x) \Rightarrow Weapon(x)$

美国的敌人是“敌对的”：

$Enemy(x, America) \Rightarrow Hostile(x)$

韦斯特上校是美国人……

$American(West)$

诺诺国是美国的敌人……

$Enemy(Nono, America)$

数据日志（datalog）知识库：数据日志是由不含函数符号的一阶确定子句组成的语言

前向链接

function FOL-FC-ASK(KB, α) **returns** 一个置换或false

inputs: KB, 知识库，一个一阶确定子句集

α, 查询，一个原子语句

while true **do**

$new \leftarrow \{\}$ //每次迭代推断出的新语句集

for each *rule* **in** KB **do**

$(p_1 \wedge \cdots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES(*rule*)

for each θ 使得对于某些KB中的$p'_1, \cdots, p'_n$有SUBST $(\theta, p_1 \wedge \cdots \wedge p_n)$ = SUBST $(\theta, p'_1 \wedge \cdots \wedge p'_n)$

$q' \leftarrow$ SUBST(θ, q)

if q'不能与已经在KB或new中的语句合一 **then**

将q'添加到new

$\phi \leftarrow$ UNIFY(q', α)

if ϕ不为*failure* **then return** ϕ

if $new = \{\}$ **then return** false

将new添加到KB

前向链接算法

前向链接

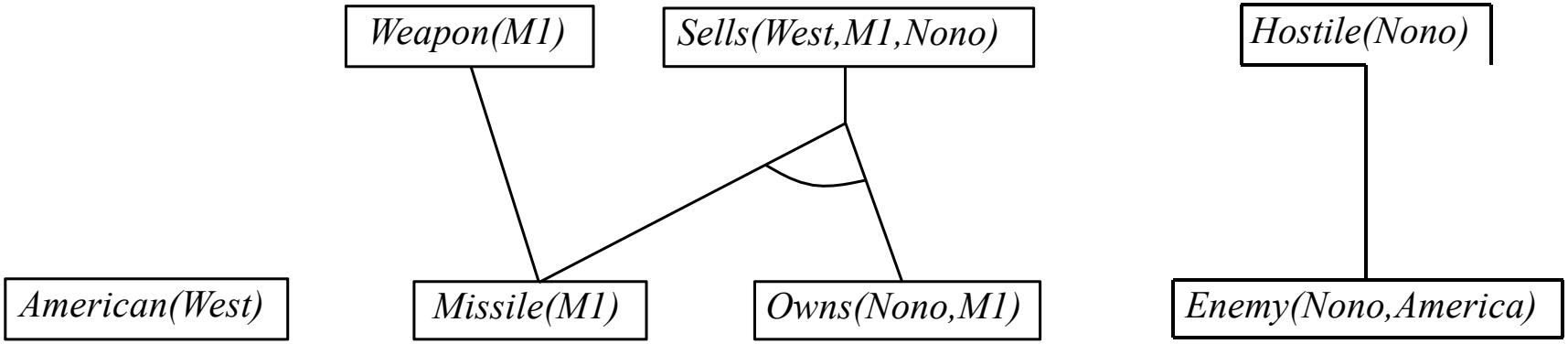
American(West)

Missile(M1)

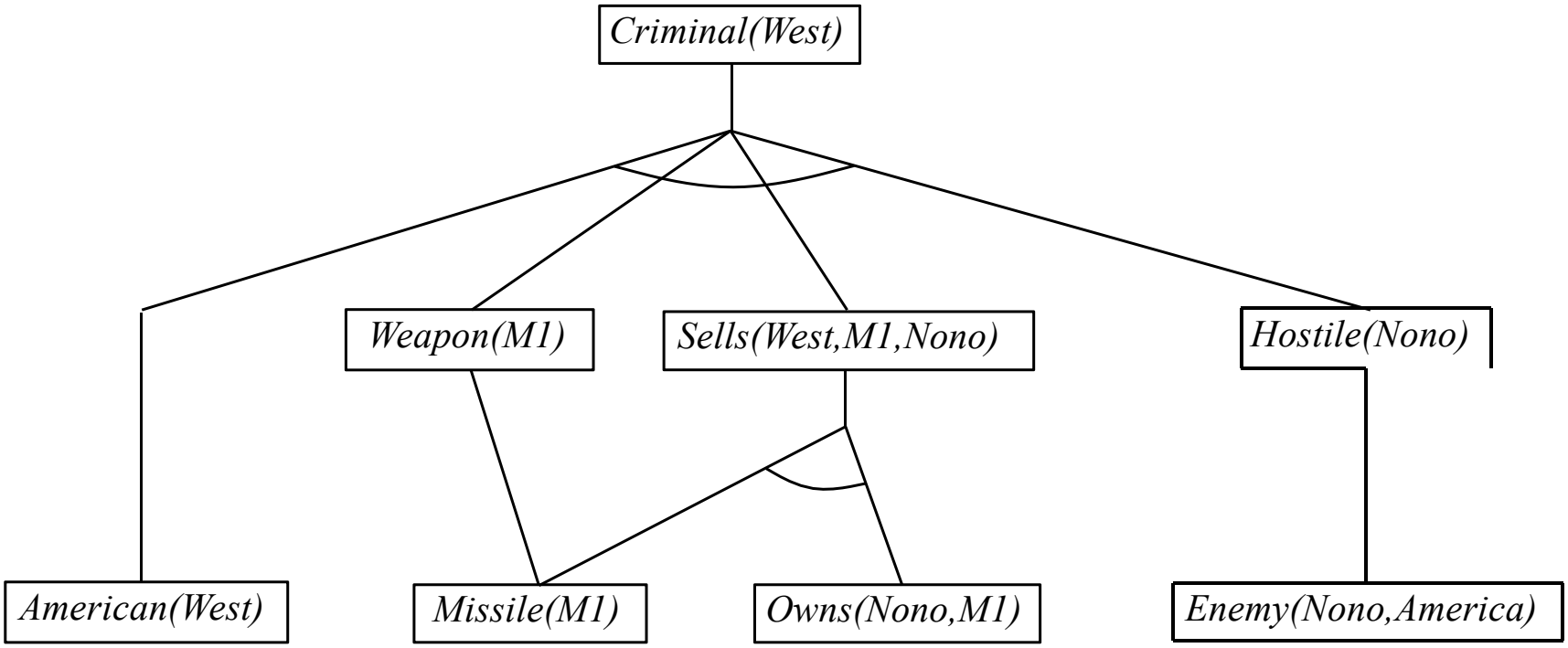
Owns(Nono,M1)

Enemy(Nono,America)

前向链接


Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)

前向链接


Criminal(West)
Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)

前向链接

前向链接的有效性

1. 将规则与已知事实进行匹配

数据库索引允许常量时间 $O(1)$ 内检索已知事实，例如，
对 $Missile(x)$ 检索 $Missile(M_1)$

2. 增量前向链接

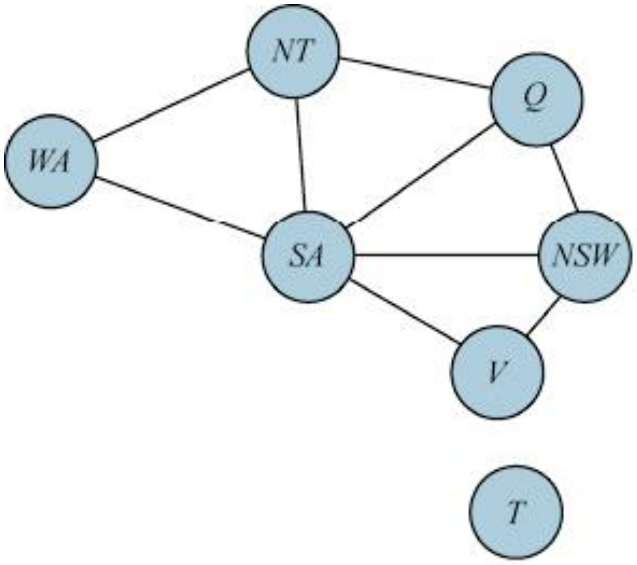
简单的观察: 在第 k 次迭代不需要匹配一条规则，
如果该规则的前件在第 $k-1$ 次迭代中没有被添加
⇒ 仅去匹配其前件包含新添加的文字的规则

3. 不相关事实

反向链接，规则子集，演绎数据库

前向链接

前向链接的有效性



（a）

$$Diff(WA, NT) \wedge Diff(WA, SA) \wedge$$
$$Diff(NT, Q) \wedge Diff(NT, SA) \wedge$$
$$Diff(Q, NSW) \wedge Diff(Q, SA) \wedge$$
$$Diff(NSW, V) \wedge Diff(NSW, SA) \wedge$$
$$Diff(V, SA) \Rightarrow Colorable()$$
$$Diff(\text{red, blue}) \quad Diff(\text{red, green})$$
$$Diff(\text{green, red}) \quad Diff(\text{green, blue})$$
$$Diff(\text{blue, red}) \quad Diff(\text{blue, green})$$

（b）

（a）用于为澳大利亚地图着色的约束图。（b）用单个确定子句表示的地图着色CSP。每个地图区域都用变量表示，变量的值可以为常量red、green、blue（使用*Diff*声明）

反向链接

```
function FOL-BC-Ask(KB, query) returns 置换生成器
  return FOL-BC-Or(KB, query, {})

function FOL-BC-Or (KB, goal, θ) returns 一个置换
  for each ruls in Fetch-Rules-For-Goal(KB, goal) do
    (lhs ⇒ rhs) ← Standardize-Variables(rule)
    for each θ' in FOL-BC-And(KB, lhs, Unify(rhs, goal, θ)) do
      yield θ'

function FOL-BC-And(KB, goals, θ) returns 一个置换
  if θ = failure then return
  else if Length(goals) = 0 then yield θ
  else
    first, rest ← First(goals), Rest(goals)
    for each θ' in FOL-BC-Or(KB, Subst(θ, first), θ) do
      for each θ'' in FOL-BC-And(KB, rest, θ') do
        yield θ''
```

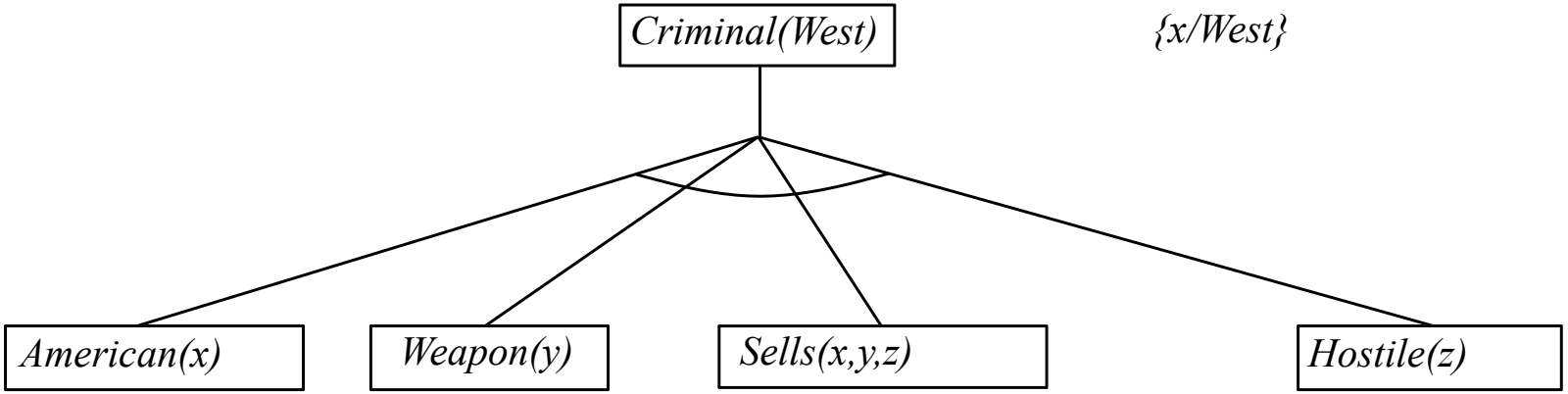
用于一阶知识库的简单的反向链接算法

反向链接

Criminal(West)

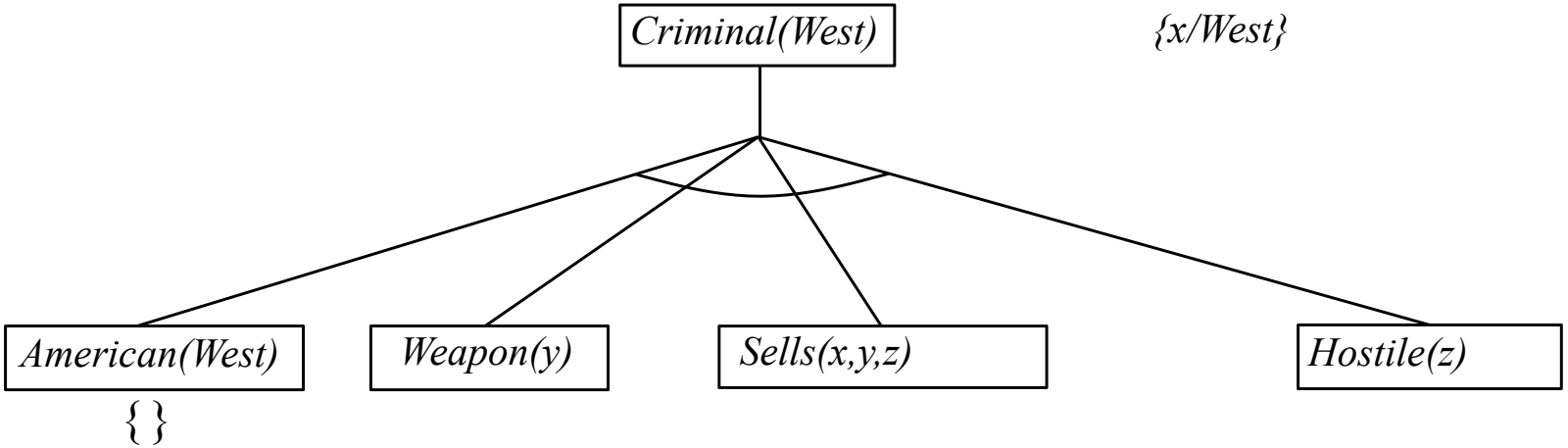
使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



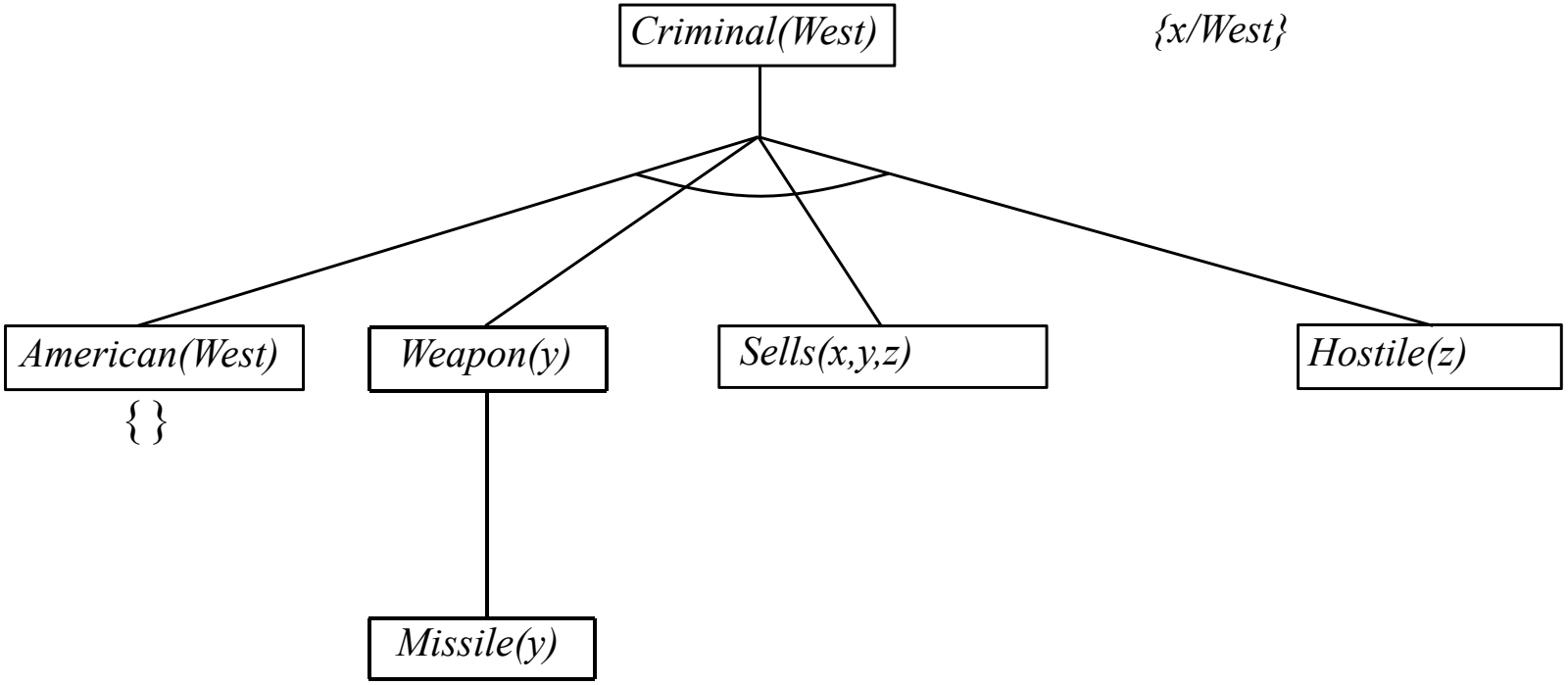
使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



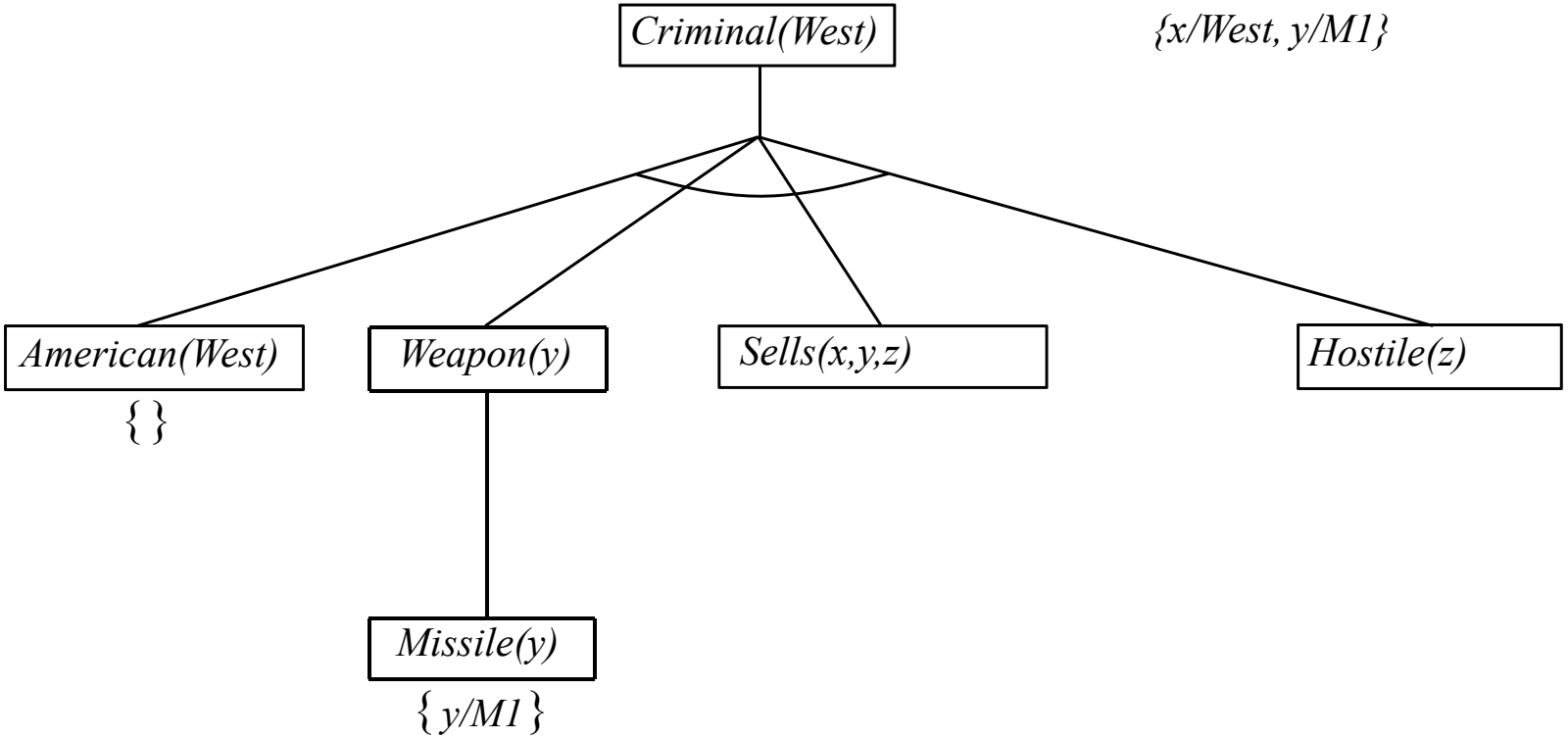
使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



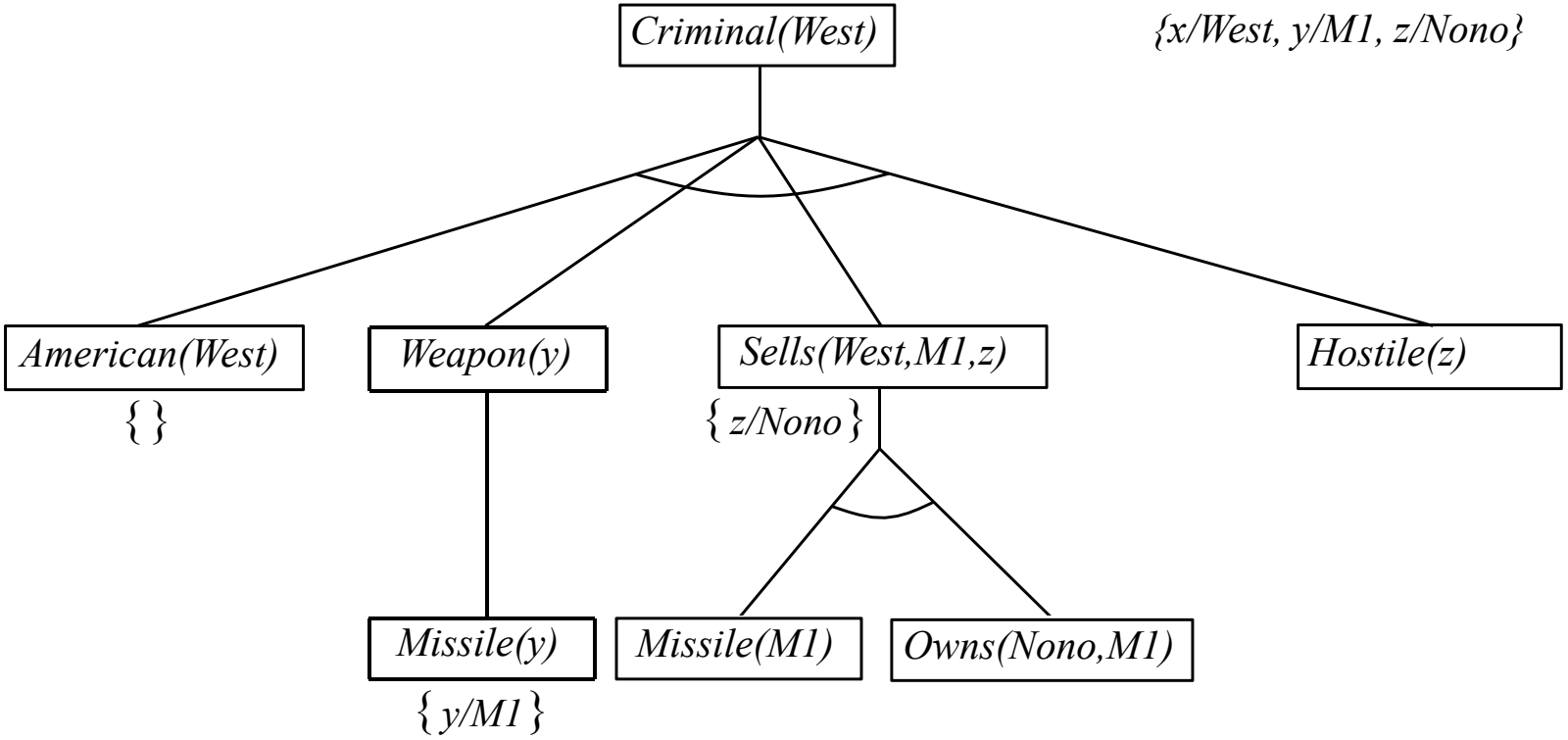
使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



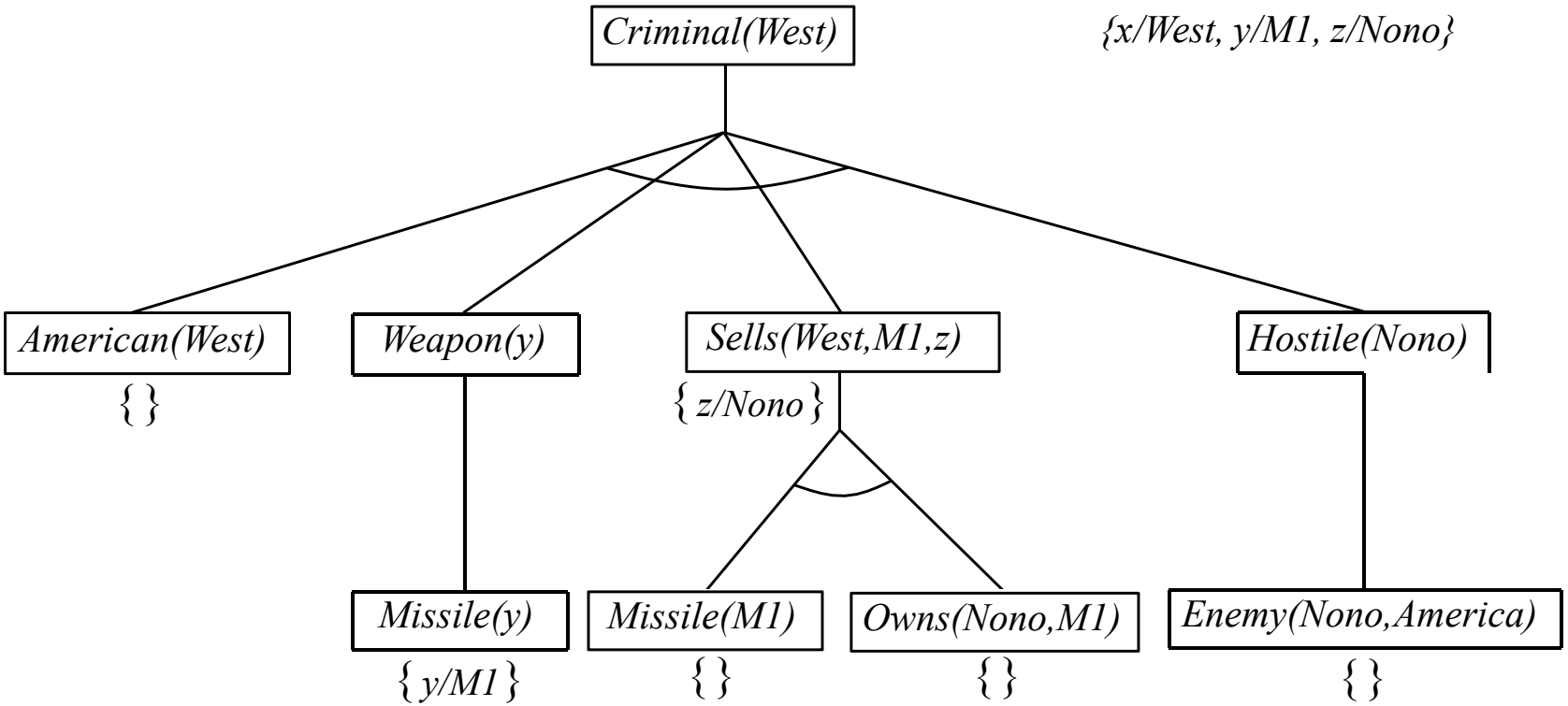
使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



使用反向链接构建证明树来证明韦斯特是有罪的

反向链接



使用反向链接构建证明树来证明韦斯特是有罪的

归结

两条进行了标准化分离、没有共同变量的子句，如果它们含有互补文字则可以被**归结**。如果两个命题文字相互否定，则这两个命题文字是互补的；如果两个一阶逻辑文字中的一个能够与另一个的否定合一，则这两个一阶逻辑文字是互补的。因此，我们有

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

其中 $\text{UNIFY}(\ell_i, \neg m_j) = \theta$

例如,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

这里 $\theta = \{x/Ken\}$

归结

南京大学 NANJING UNIVERSITY

一阶逻辑的合取范式

转化为合取范式（CNF）的步骤如下：

Everyone who loves all animals is loved by someone:

$$\forall x \ [\forall y \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y \ Loves(y, x)]$$

1. 蕴涵消去

$$\forall x \ [\neg\forall y \ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y \ Loves(y, x)]$$

2. ¬内移: $\neg\forall x \ p \equiv \exists x \ \neg p$, $\neg\exists x \ p \equiv \forall x \ \neg p$:

$$\forall x \ [\exists y \ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y \ Loves(y, x)]$$
$$\forall x \ [\exists y \ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y \ Loves(y, x)]$$
$$\forall x \ [\exists y \ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y \ Loves(y, x)]$$

归结

南京大学 NANJING UNIVERSITY

一阶逻辑的合取范式

转化为合取范式（CNF）的步骤如下：

3. 变量标准化：

   $\forall x \ [\exists y \ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z \ Loves(z, x)]$

4. 斯科伦化：通过消去移除存在量词。此处F和G为斯科伦函数（Skolem function），其参数全部是全称量化的变量，要消去的存在量词出现在这些变量的辖域内：

   $\forall x \ [Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

5. 全称量词消除：

   $[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

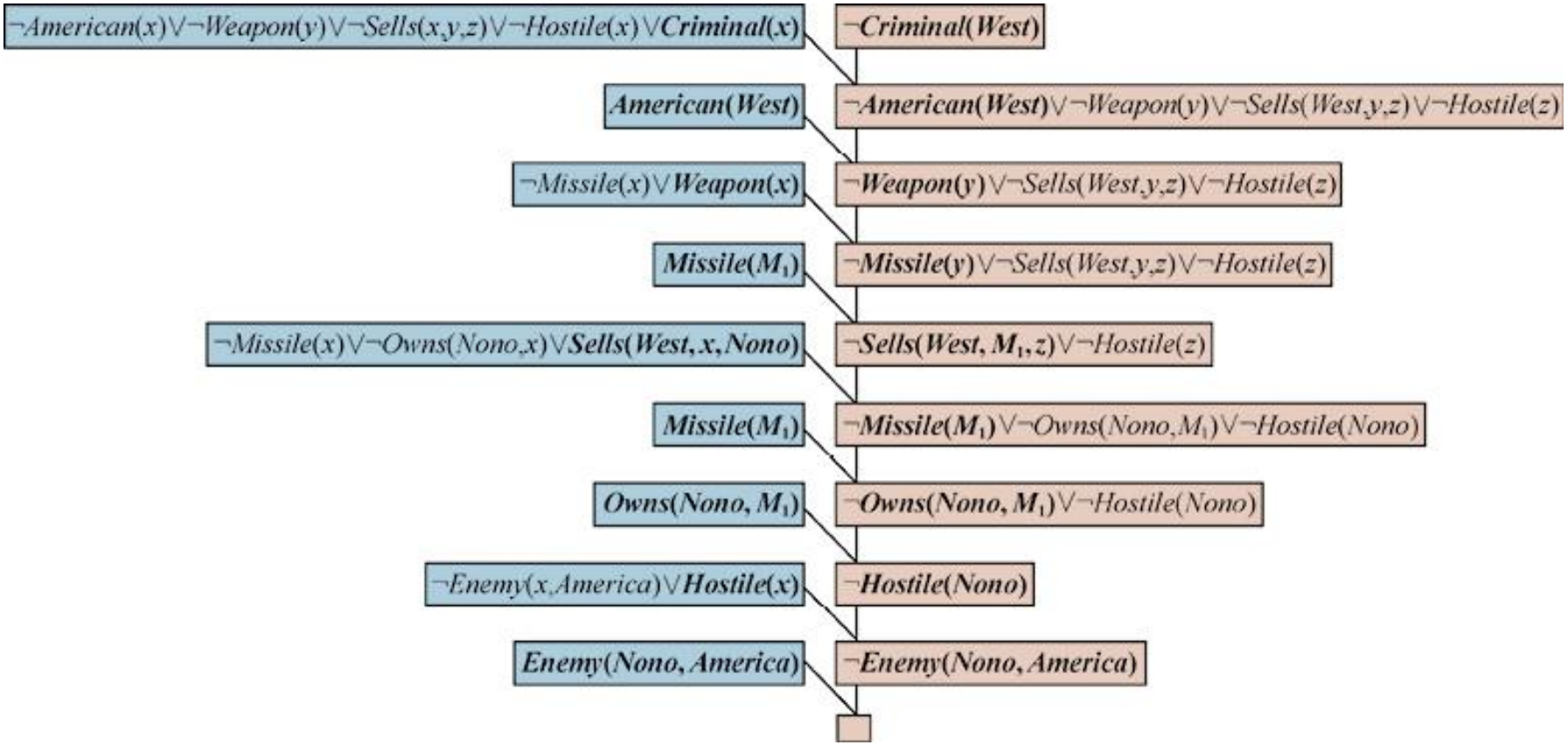
6. 对 $\wedge$ 分配 $\vee$：

   $[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

归结

南京大学 NANJING UNIVERSITY

归结证明范例



归结证明韦斯特有罪。每个归结步骤中，合一文字用加粗字体表示，带有正文字的子句用蓝底表示。

小结

- 使用推断规则（**全称量词实例化**和**存在量词实例化**）来**命题化**一阶逻辑的推断问题。通常，这种方法速度慢，除非论域非常小。
- 使用**合一**来找出适当的变量置换消去一阶证明中的实例化步骤，在许多情况下提高了这一过程的效率。
- 肯定前件的提升版使用了合一，产生了**一般化肯定前件**这种自然、强大的推断规则。**前向链接**算法和**反向链接**算法对确定子句集使用这条规则。
- 一般化肯定前件对确定子句是完备的，但蕴含问题是**半可判定的**。对于不含函数的确定子句构成的数据日志知识库，蕴含是可判定的。
- 前向链接用于**演绎数据库**，此时它可以与关系数据库的操作结合。它也用于对超大规则集进行高效更新的产生式系统。前向链接对于数据日志是完备的，并可以在多项式时间内运行。
- 反向链接用于**逻辑编程系统**，它利用巧妙的编译器技术来实现超快速推断。反向链接受制于冗余推断和死循环，可以通过备忘来缓解这些问题。
- 一般化的**归结**推断规则使用合取范式知识库为一阶逻辑提供了完备的推断系统。

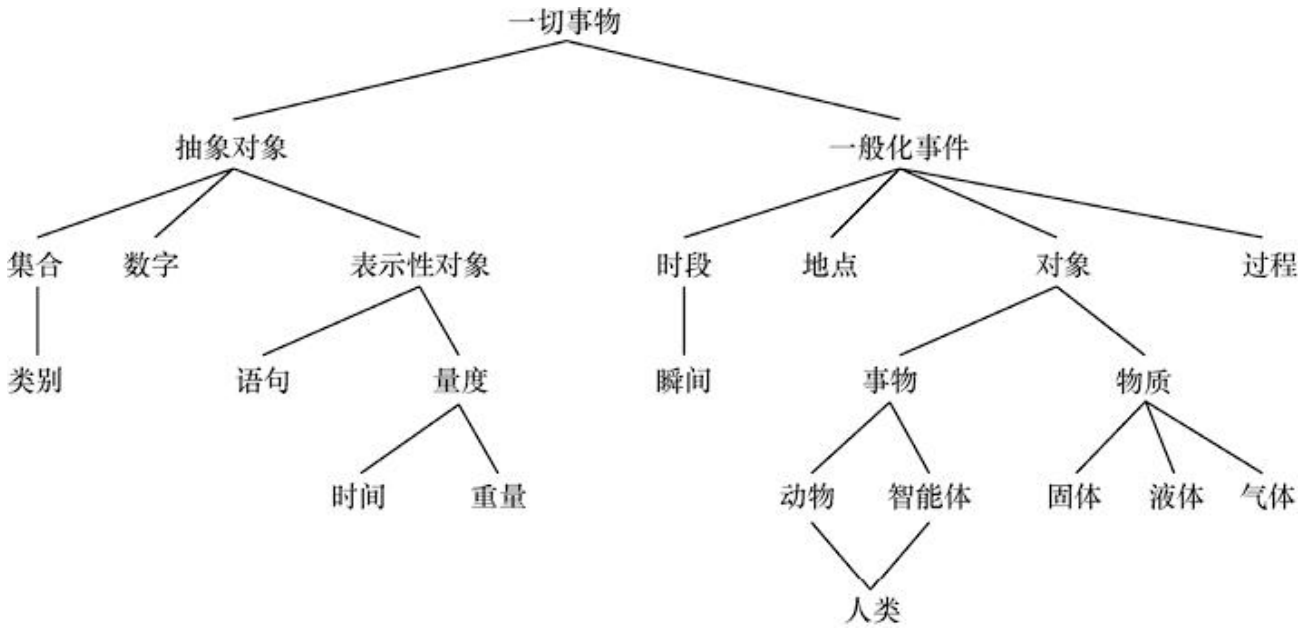
提纲

南京大学 NANJING UNIVERSITY

- 一阶逻辑中的推断
  - 命题推断与一阶推断
  - 合一与一阶推断
  - 前向链接与反向链接
  - 归结
- **知识表示**
- 自动规划
  - 经典规划
  - 分层规划

本体论工程

- **本体论工程（ontological engineering）**：对抽象概念进行表示，如事件、时间、对象、信念等。
- 概念的一般性的框架被称为**上层本体论（upper ontology）**，因为通常将更一般性的概念绘制于更具体的概念之上。



世界的上层本体论

类别与对象

- 将对象组织为类别是知识表示的重要组成部分。
- 用一阶逻辑表示类别有两种选择：谓词和对象。
- 类别通过继承组织知识。

类别与对象

- 将对象组织为类别是知识表示的重要组成部分。
- 用一阶逻辑表示类别有两种选择：谓词和对象。
- 类别通过继承组织知识。
- **物理组成**：一个对象是另一个对象的一部分
  - 例如: 罗马尼亚是欧洲的一部分： $PartOf(Romania, Europe)$

类别与对象

南京大学 NANJING UNIVERSITY

- 将对象组织为类别是知识表示的重要组成部分。
- 用一阶逻辑表示类别有两种选择：谓词和对象。
- 类别通过继承组织知识。
- **物理组成：**一个对象是另一个对象的一部分
  - 例如: 罗马尼亚是欧洲的一部分： $PartOf(Romania, Europe)$
- **量度：**我们分配给对象属性的值。**可被排序。**
  - 例如: $Length(L_1) = Inches(1.5) = Centimeters(3.81)$

类别与对象

- **物质**: 不能被个体化（individuation）。例如，无法说出“黄油对象”的明确数量，因为一个黄油对象的任何一部分也是黄油对象。
- **固有的（intrinsic）**：属于对象的物质，将一种具体的物质切成两半，这两半仍保留其固有性质，如其密度、沸点、味道、颜色、隶属关系等。
- **外在的（extrinsic）**：如重量、长度、形状之类的，在分割后并不能维持不变。
- 定义中只有固有性质的对象类别是物质，或不可数名词；而定义中含有任何外在性质的类别则是可数名词。

事件

- **事件演算**: 事件、流和时间点
  - 流，例如，*At(Shankar, Berkeley)*
  - 事件，例如，事件 E_1：Shankar 从旧金山飞往华盛顿特区
  - $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge Origin(E_1, SF) \wedge Destination(E_1, DC)$
  - *Flyings* 是所有飞行事件的类别。

- 事件演算的谓词集

$T(f, t_1, t_2)$	流 f 在 t_1 和 t_2 之间的所有时刻为真
$Happens(e, t_1, t_2)$	事件 e 从 t_1 开始，于 t_2 结束
$Initiates(e, f, t)$	事件 e 导致流 f 在时刻 t 为真
$Terminates(e, f, t)$	事件 e 导致流 f 在时刻 t 不再为真
$Initiated(f, t_1, t_2)$	流 f 在 t_1 和 t_2 之间的某时刻开始为真
$Terminated(f, t_1, t_2)$	流 f 在 t_1 和 t_2 之间的某时刻停止为真
$t_1 < t_2$	时刻 t_1 出现在时刻 t_2 之前

- 我们可以将飞行事件的效果描述为：

$$E = Flyings(a, here, there) \wedge Happens(E, t_1, t_2) \Rightarrow$$
$$Terminates(E, At(a, here), t_1) \wedge Initiates(E, At(a, there), t_2)$$

精神对象和模态逻辑

- **精神对象**是智能体大脑中（或知识库中） 的知识
- 智能体对精神对象具有**命题态度（Propositional attitudes）**，例如，相信、知道、想要、通知。
- 假设我们试图断言露易丝知道超人会飞：$Knows(Lois, CanFly(Superman))$，通过等值推理可得：

$$(Superman = Clark) \wedge Knows(Lois, CanFly(Superman)) \models Knows(Lois, CanFly(Clark))$$

- **模态逻辑**含有以语句（而非项）为参数的**模态算子**（modal operator）。
- “A 认识 P”被表示为 $\boldsymbol{K}_A P$, 这里 $\boldsymbol{K}$ 是知识的模态算子，它使用两个参数，一个是智能体（以下标表示），另一个是语句。

精神对象和模态逻辑

南京大学 NANJING UNIVERSITY

- 智能体能够进行推论如果一个智能体知道P且知道P蕴涵Q，则该智能体知道Q:

$$(\boldsymbol{K}_a P \wedge \boldsymbol{K}_a(P \Rightarrow Q)) \Rightarrow \boldsymbol{K}_a Q$$

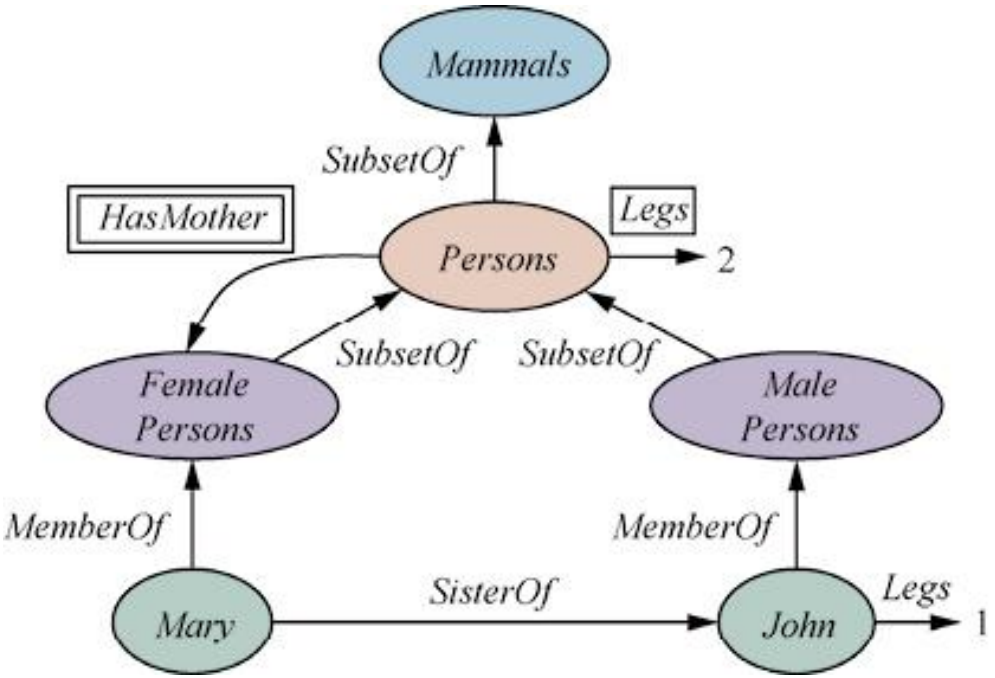
- 逻辑智能体（但并非所有人）都能够内省自己的知识。如果它们知道某事，则它们知道“它们知道这事”：

$$\boldsymbol{K}_a P \Rightarrow \boldsymbol{K}_a (\boldsymbol{K}_a P)$$

类别的推理系统

语义网络

语义网络的记法便于进行继承推理。



具有4个对象（*John*、*Mary*、1和2）和4个类别的语义网络。关系使用带标签的连线表示。

类别的推理系统

南京大学 NANJING UNIVERSITY

描述逻辑

- 旨在简化对类别的定义和性质的描述的记法。
- 描述逻辑系统是从语义网络演化而来的，目的是在维持分类结构作为组织原则的同时形式化网络的含义。
- 主要推断任务:
  - **包容**: 通过比较定义来检查一个类别是否是另一个类别的子集
  - **分类**: 检查一个对象是否属于某个类别
- CLASSIC 语言 (Borgida et al., 1989)是典型的描述逻辑
  - 例如:单身汉是未婚成年男性

$$bachelor = And(Unmarried, Adult, Male)$$

  - 一阶逻辑中的等价形式

$$Bachelor(x) \Leftrightarrow Unmarried(x) \wedge Adult(x) \wedge Male(x)$$

用缺省信息推理

南京大学 NANJING UNIVERSITY

限定与缺省逻辑

用缺省信息推理

南京大学 NANJING UNIVERSITY

限定与缺省逻辑

- **限定**可以被看作更为强大且精确的封闭世界假设。
  - 确定一个被假设为“尽可能假”的谓词
  - 作一种模型偏好（model preference）逻辑
  - 例如，假设我们要断言缺省规则，鸟会飞。

$$Bird(x) \wedge \neg Abnormal_1(x) \Rightarrow Flies(x)$$

用缺省信息推理

南京大学 NANJING UNIVERSITY

限定与缺省逻辑

- **限定**可以被看作更为强大且精确的封闭世界假设。
  - 确定一个被假设为“尽可能假”的谓词
  - 作一种模型偏好（model preference）逻辑
  - 例如，假设我们要断言缺省规则，鸟会飞。

$$Bird(x) \wedge \neg Abnormal_1(x) \Rightarrow Flies(x)$$

- **缺省逻辑**是可以用缺省规则生成逻辑偶然的非单调结论的形式体系。
  - 例如: $Bird(x) : Flies(x) / Flies(x)$
  - 这条规则的意思是，如果$Bird(x)$为真，且如果Flies(x)与知识库一致，则可以得出缺省结论$Flies(x)$。

用缺省信息推理

南京大学 NANJING UNIVERSITY

真值维护系统

- **信念修正**: 在有新信息的情况下，许多推断出的事实最终将被证明是错误的，必须被收回。

- **真值维护系统（TMS）**,旨在处理从错误语句推断出的任何其他语句的复杂情况。

- **基于论证的真值维护系统（JTMS)**
  - 知识库中的每条语句都用一条论证（justification）标记，它含有推断出该语句的语句集。
  - 论证使收回语句变得高效。
  - 假设被考虑过一次的语句很可能会再次被考虑。

提纲

- 一阶逻辑中的推断
  - 命题推断与一阶推断
  - 合一与一阶推断
  - 前向链接与反向链接
  - 归结
- 知识表示
- **自动规划**
  - 经典规划
  - 分层规划

经典规划

- **经典规划（classical planning）**定义为在一个离散的、确定性的、静态的、完全可观测的环境中，找到完成目标的一系列动作的任务。

- **规划领域定义语言（Planning Domain Definition Language, PDDL）**是一种因子化语言。
  - 基本的PDDL可以处理经典规划领域，而扩展的PDDL则可以处理连续的、部分可观测的、并发的和多智能体的非经典领域。
  - **状态**表示为基本原子流的合取。
  - 使用**数据库语义**：封闭世界假设意味着没有提到的任何流都是假的。

经典规划

- 一个**动作模式（action schema）**表示一组基本动作。例如，下面是一个使飞机从一个位置飞到另一个位置的动作模式：

$$Action(Fly(p, from, to),$$

$$\text{PRECOND}: At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$$

$$\text{EFFECT}: \neg At(p, from) \wedge At(p, to))$$

- 模式由动作名称、模式中使用的所有变量的列表、**前提**（precondition）和**效果**（effect）组成。

- 一组动作模式是规划领域的一个定义。领域中的特定问题通过添加初始状态和目标来定义。

- **初始状态**是基本流的合取。

- **目标**（使用Goal引入）和前提类似，它是可以含有变量的文字（正文字或负文字）合取式。

经典规划

范例领域：航空货物运输

一个解序列：$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$
$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)]$

$Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)$
$\land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)$
$\land Airport(JFK) \land Airport(SFO))$
$Goal(At(C_1, JFK) \land At(C_2, SFO))$
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $\neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
EFFECT: $\neg At(p, from) \land At(p, to))$

航空货物运输规划问题的PDDL描述

经典规划

范例领域：备用轮胎问题

一个解序列：$[Remove(Flat, Axle), Remove(Spare, Trunk), PutOn$

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$

$Goal(At(Spare, Axle))$

$Action(Remove(obj, loc),$

PRECOND: $At(obj, loc)$

EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$

$Action(PutOn(t, Axle),$

PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle) \land \neg At(Spare, Axle)$

EFFECT: $\neg At(t, Ground) \land At(t, Axle))$

$Action(LeaveOvernight,$

PRECOND:

EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
$\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

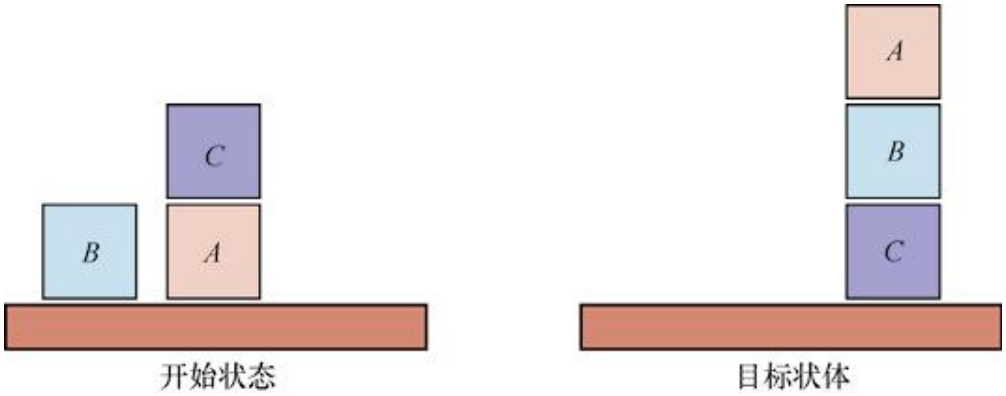
备用轮胎问题的PDDL描述

经典规划

范例领域：积木世界

一个解序列：$[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$

积木世界问题的示意图

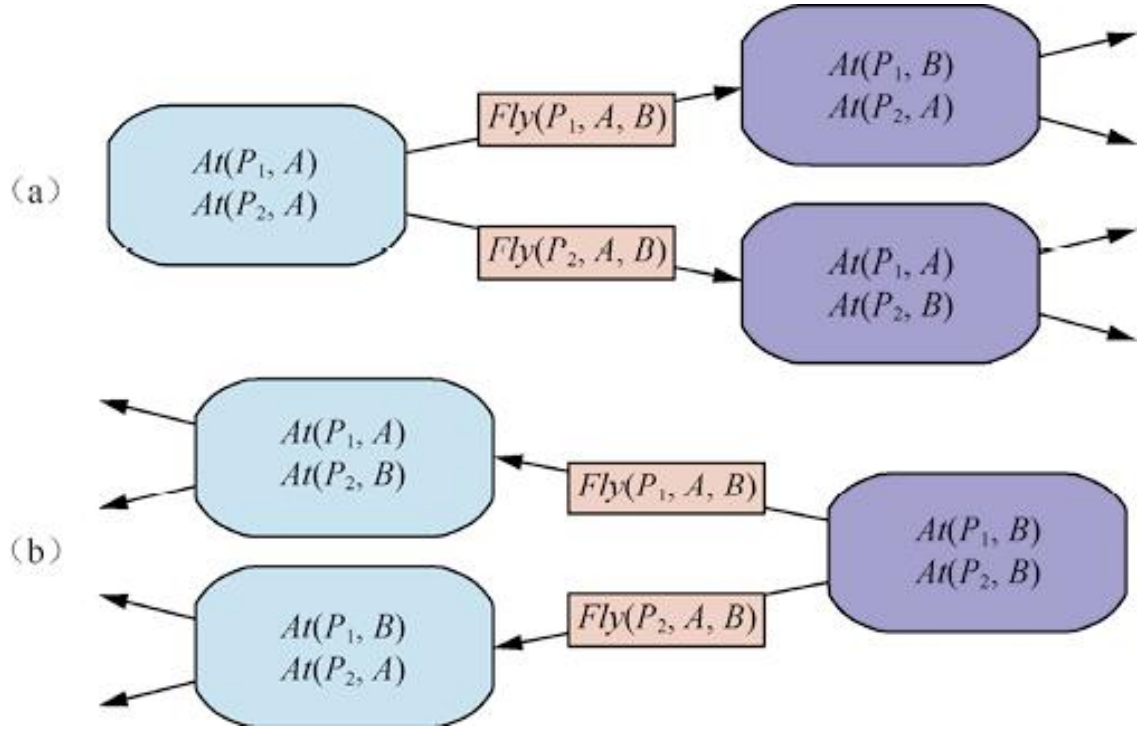


积木世界问题的PDDL描述。

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A)$
$\wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C) \wedge Clear(Table))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge$
$(b \neq x) \wedge (b \neq y) \wedge (x \neq y)$
EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge Block(x),$
EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

经典规划

经典规划的算法



搜索规划的两种方法。（a）在基本状态空间中**前向（递进）搜索**，从初始状态开始，利用问题的动作向前搜索目标状态集合中的一个成员。（b）通过状态描述进行**反向（回归）搜索**，从目标开始，用逆动作反向搜索初始状态。

经典规划

经典规划的算法

- **规划的前向状态空间搜索**
  - 开始于初始状态
  - 为了确定**适用动作**，我们将当前状态与当每个动作模式的前提合一。
  - 对于每个成功产生置换的合一，我们将该置换应用于动作模式来产生不含变量的基本动作。

- **规划的反向状态空间搜索**
  - 从目标开始反向应用动作，直到找出到达初始状态的步骤序列。
  - 在每一步中，我们考虑**相关动作。**
  - **极大地减少了分支因子。**
  - 相关动作是其效果能够与目标中的一个文字合一的动作，但这个效果不否定目标的任何部分的行动。

经典规划

经典规划的算法

- **使用布尔可满足性规划**：基于SAT的规划器SATPlan通过将PDDL问题描述翻译为命题形式来工作。步骤包括：
  - 将动作命题化
  - 添加动作排除公理来表明两个动作不能同时发生
  - 添加前提公理
  - 定义初始状态
  - 将目标命题化
  - 添加后继状态公理

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$

规划的启发式方法

南京大学 NANJING UNIVERSITY

- **忽略前提启发式方法：**从动作中去掉所有的前提。
  - 每一个动作都适用于所有状态

- **忽略删除列表启发式方法：**从所有动作中移除删除列表（例如，从效果中移除所有负文字）。任何动作都不会抵消另一个动作所取得的进展。

- **领域无关剪枝**
  - **对称约简：**修剪掉搜索树中的所有对称分支而不予考虑，只保留其中一个。
  - **松弛规划**: 松弛问题的求解
  - **优先动作**:松弛规划中的一步，或者能够满足松弛规划的某个前提

规划的启发式方法

南京大学 NANJING UNIVERSITY

- **规划中的状态抽象**
  - **状态抽象**：从问题的基本表示到抽象表示的多对一映射
  - **分解**：将问题分解为多个部分，独立求解各个部分，然后将这些部分组合起来
  - **子目标独立假设**：求解一系列子目标的代价近似于独立求解每个子目标的成本之和
  - 子目标独立性假设可以是乐观的，也可以是悲观的
    - **乐观的**：当每个子目标的子规划之间存在负相互作用
    - **悲观的**：不可容许的，子规划包含冗余的动作

分层规划

南京大学 NANJING UNIVERSITY

- 为了控制复杂性，使用层次分解进行更高层次的抽象
- 分层的每一层级都将计算任务减少为下一层次的少量活动，因此找出为当前问题安排这些活动的正确方法的计算代价很小。
- **高层动作 (HLA):**
  - 分层任务网络领域规划
  - 假设完全可观测性和确定性以及**基元动作**具有标准的前提-效果模式
  - 每个HLA都有一个或多个可能的细化（refinement）来形成一个动作序列
  - HLA的实现：只包含基元动作的HLA细化.
- **高层规划 (HLP)**
  - 连接序列中每个HLA的实现
  - 如果一个高层规划至少有一个实现从给定状态达成了目标，则该高层规划从该给定状态达成了目标

分层规划

细化示例

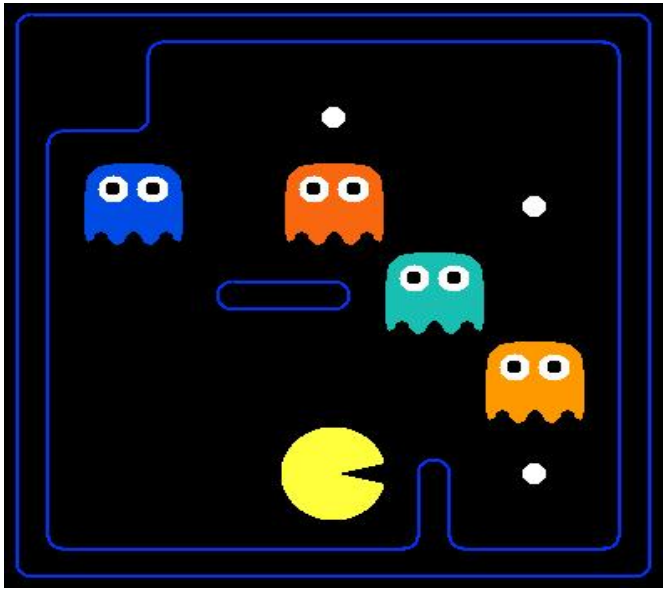
Refinement(*Go*(*Home*, *SFO*),
  STEPS: [*Drive*(*Home*, *SFOLongTermParking*),
    Shuttle(*SFOLongTermParking*, *SFO*)])
Refinement(*Go*(*Home*, *SFO*),
  STEPS: [*Taxi*(*Home*, *SFO*)])

Refinement(*Navigate*([*a*, *b*], [*x*, *y*]),
  PRECOND: $a = x \wedge b = y$
  STEPS: [])
Refinement(*Navigate*([*a*, *b*], [*x*, *y*]),
  PRECOND:*Connected*([*a*, *b*], [$a - 1$, *b*])
  STEPS: [*Left*, *Navigate*([$a - 1$, *b*], [*x*, *y*])])
Refinement(*Navigate*([*a*, *b*], [*x*, *y*]),
  PRECOND:*Connected*([*a*, *b*], [$a + 1$, *b*])
  STEPS: [*Right*, *Navigate*([$a + 1$, *b*], [*x*, *y*])])
...

两个高层动作——前往旧金山机场和在真空吸尘器世界中导航——的可能细化的定义。注意真空吸尘器世界中细化的递归特性和对前提的使用。

课程作业2：逻辑与经典规划

在这次课程作业中，你将使用/编写Python函数来生成逻辑语句用以描述吃豆人世界，包括使用 Expr 数据类型来表示和实现命题逻辑语句和函数，以及使用 SAT（可满足性问题）求解器 pycosat 来**解决与规划相关的逻辑推断任务**，例如，生成动作序列以到达目标位置并吃掉所有豆子。



课程作业2：逻辑与经典规划

南京大学 NANJING UNIVERSITY

基于命题逻辑的智能体

- 逻辑智能体通过用关于世界的语句**知识库**推导接下来的动作来运作。知识库由**公理（axiom）**，也就是关于世界如何运行的一般知识，和从智能体在某个特定世界获得的**感知**语句构成。

- **构建知识库**
  - **收集公理**，例如，在wumpus世界中起始方格没有无底洞，恰好只有一只wumpus，等等。
  - **感知**，例如，使用S_1_1表示位置[1,1]有臭味，对于世界随时间变化的部分（**流**）还需引入时间 t。
  - **转移模型：后继状态公理（successor-state axiom）。**对于每个流 F，$F\_(t+1)$ 的真值可以用两种方法之一来确定：一种是时刻 t 的动作导致 F 在 $t+1$ 为真，另一种是 F 在时刻 t 已经为真而时刻 t 的动作没有导致它为假，即，

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$

- 给定完整的后继状态公理和之前列出的其他公理，智能体就能够询问和回答**世界当前状态**的所有可解答问题。

课程作业2：逻辑与经典规划

南京大学 NANJING UNIVERSITY

基于命题逻辑的智能体

- **用命题推断进行规划**
  - 构建一个语句，它含有：
    a. 对于初始状态的断言集
    b. 到最大为时刻 t 为止的每一时间步的所有可能动作的后继状态公理
    c. 目标在时刻 t 达成的断言
  - 将所有语句提供给SAT求解器。如果求解器找到一个可满足的模型，则目标是可达成的；如果语句不可满足，则问题无解。
  - 假设找到了一个模型，从模型中提取代表动作并被赋值为true的变量。它们代表一个达成目标的规划。

课程作业2：逻辑与经典规划

南京大学 NANJING UNIVERSITY

基于命题逻辑的智能体

- **用命题推断进行规划**

function SATPLAN(*init*, *transition*, *goal*, T_{max}) **returns** 解或 *failure*
  inputs: *init*, *transition*, *goal*，它们构成了问题的描述
    T_{max}，规划的长度上限

  for $t = 0$ **to** T_{max} **do**
    cnf ← TRANSLATE-TO-SAT(*init*, *transition*, *goal*, *t*)
    model ← SAT-SOLVER(*cnf*)
    if *model*非空 **then**
      return EXTRACT-SOLUTION(*model*)
  return *failure*

SATPlan算法。规划问题被转换为CNF语句，其中目标被断言在固定的时间步t时成立，到t为止的每个时间步都含有公理。如果可满足性算法找到了一个模型，则通过查看指向动作并在模型中被赋值为true的命题符号来提取规划。如果模型不存在，则将目标后移一步，重复这一过程。

课程作业2：逻辑与经典规划

南京大学 NANJING UNIVERSITY

基于命题逻辑的吃豆人智能体

- **地图**：哪里有围墙以及哪里没有围墙
- **初始状态**：吃豆人位于地图的某处
- **域约束**：
  - 吃豆人在每一步只采取一个动作
  - 吃豆人在每一步只位于一个位置
- **感知模型**： <Percept_t> ⇔ <some condition on world_t>
- **转移模型**：
  - <at x,y_t> ⇔ [at x,y_t-1 and stayed put] v [next to x,y_t-1 and moved to x,y]