COMP4418 Review Final B

Lecturer: Yihan Xiao

Outline

- Proposition Logic
- First Order Logic
- Resolution & SLD
- Knowledge Representation
- Reasoning
- Answer Set Programming
- Satisfiability
- Knowledge and Action

Proposition Logic 命题逻辑

命题的概念
一个命题是一个非真即假的陈述句。

简单命题（原子命题）

简单命题只对一个事物的一个性质进行判断

Eg. 雪是白的。
Eg. 我下午在图书馆。

复合命题

从语法结构上可分解成若干简单命题的命题是复合命题

Eg. 我下午在图书馆，或者去打球。
Eg. 如果明天不下雨，我们就去旅行。

Proposition Logic 命题逻辑

命题的符号化表示

一个命题常量是一个表达了具体的命题内容的命题，可使用一个形式符号

P：张三是大学生

逻辑符号

1. 否定¬： 设 P 为一个命题，则 ¬P 也为一个命题，称为 P 的否定

P：我喜欢4418这门课
¬P： “我喜欢4418这门课” 是假的

P	¬P
F	T
T	F

2. 合取词 ∧：

P：今天是星期六。
Q：雪是黑的。
P∧Q：今天是星期六而且雪是黑的。

P	Q	P∧Q
F	F	F
F	T	F
T	F	F
T	T	T

逻辑符号

3. 析取词 ∨

P：同学们在晚会上唱歌。
Q：同学们在晚会上跳舞。
P∨Q：同学们在晚会上载歌载舞。

P	Q	P∨Q
F	F	F
F	T	T
T	F	T
T	T	T

Tip: 析取是不互斥的

P：今天下午5点我在图书馆。
Q：今天下午5点我在足球场。
如何表达：今天下午5点我在图书馆或足球场。

P∨Q ？

Ans: (P∧¬Q)∨(¬P∧Q)

逻辑符号

4. 蕴含→

设 P、Q 为命题，则 P → Q 也是一个命题，称为 P 条件蕴含 Q

P称为逻辑前件，Q称为逻辑后件。
一般P → Q 可化为¬ P ∨ Q

与自然语言的对应：如果…那么…，意味着，蕴含…

P：张三是大学生。
Q：张三成绩很好。
P→Q：如果张三是大学生，那么张三成绩很好。

P	Q	P→Q
F	F	T
F	T	T
T	F	F
T	T	T

5. 双重蕴含↔

设 P、Q 为命题，则 P ↔ Q 也是一个命题，称为 P 双重蕴含 Q

与自然语言的对应：…当且仅当…，充要条件

一般P ↔ Q 可化为(P → Q) ∧ (Q → P)

P：张三是大学生。
Q：张三成绩很好。
P ↔ Q：张三是大学生当且仅当张三成绩很好。

P	Q	P↔Q
F	F	T
F	T	F
T	F	F
T	T	T

范式 Normal Forms

否定范式 Negation Normal Form

取反符号只出现在原子命题之前 $(P \vee \neg Q) \rightarrow (P \wedge (\neg R \vee S))$

合取范式 Conjunctive Normal Form (CNF)

所有子命题以逻辑与(交集)的形式相连接

$$(P \vee Q \vee \neg R) \wedge (\neg S \vee \neg R))$$

析取范式 Disjunctive Normal Form (DNF)

所有子命题以逻辑或(并集)的形式相连接

$$(P \wedge Q \wedge \neg R) \vee (\neg S \wedge \neg R))$$

合取范式 Conjunctive Normal Form (CNF)

任何命题都可以转化为CNF的表达形式，主要有三种规则

1. 蕴含转化

A (sub)formula $\phi \to \psi$ is equivalent to $\neg\phi \vee \psi$

A (sub) formula $\phi \leftrightarrow \psi$ is equivalent to $\phi \to \psi$ and $\psi \to \phi$

2. 第摩根法则

DeMorgan's laws:

- $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$
- $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

3. 分配率

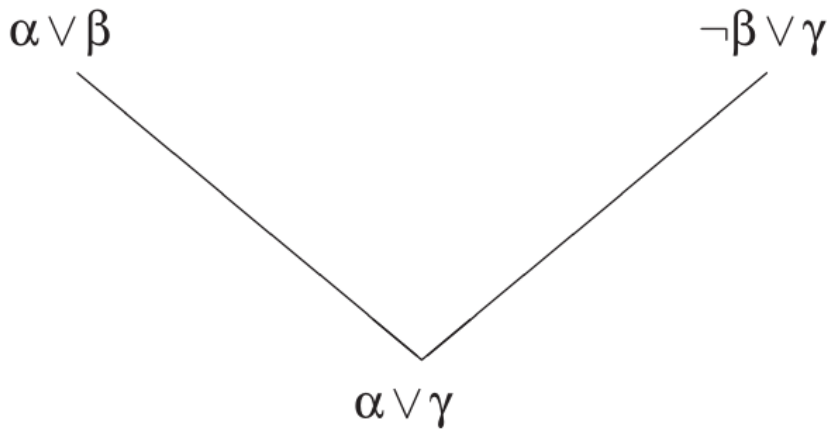
Note the following distributive identities:

$$(\phi \wedge \psi) \vee \chi \equiv (\phi \vee \chi) \wedge (\psi \vee \chi)$$
$$(\phi \vee \psi) \wedge \chi \equiv (\phi \wedge \chi) \vee (\psi \wedge \chi)$$

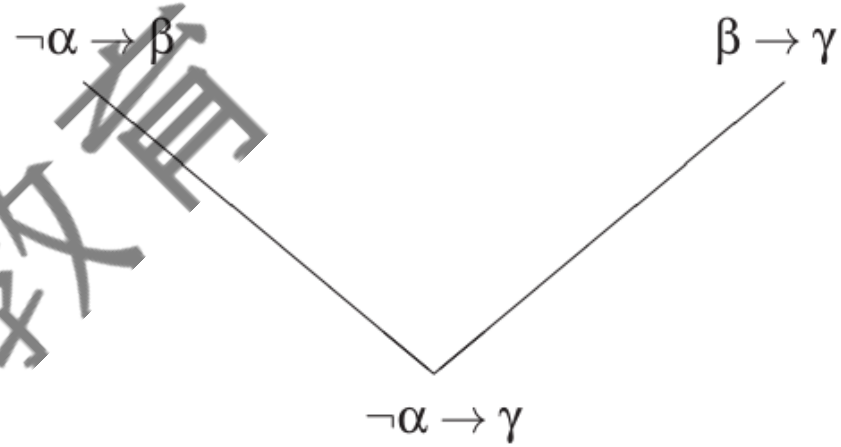
命题逻辑的推理

Resolution Rule

1. 矛盾项抵消



2. 逻辑承接



命题逻辑的推理

Refutation Systems

将要证明命题的否定形式加入到已知命题集合中，看能否推出矛盾，即反证法

为何要使用Refutation system?

具体步骤：

1. 将已知命题（前提）和要证明命题的否定形式转化为CNF
2. 重复使用推理定律进行推理
3. 看是否能推出矛盾 empty clause

命题逻辑的推理

$$(G \vee H) \rightarrow (\neg J \wedge \neg K),\ G \vdash \neg J$$

1. 转化为CNF

$$CNF[(G \vee H) \rightarrow (\neg J \wedge \neg K)] \equiv (\neg G \vee \neg J) \wedge (\neg H \vee \neg J) \wedge (\neg G \vee \neg K) \wedge (\neg H \vee \neg K)$$

2. 重复进行推理

1. $\neg G \vee \neg J$ [Premise]
2. $\neg H \vee \neg J$ [Premise]
3. $\neg G \vee \neg K$ [Premise]
4. $\neg H \vee \neg K$ [Premise]
5. G [Premise]
6. $\neg\neg J$ [¬ Conclusion]
7. J [6. Double Negation]
8. $\neg G$ [1, 7. Resolution]

3. 得到矛盾，得证

9. $\square$ [5, 8. Resolution]

命题逻辑的推理

前提为空时也可以使用Refutation $\vdash ((P \vee Q) \wedge \neg P) \rightarrow Q$

1. 转化为CNF $CNF[\neg(((P \vee Q) \wedge \neg P) \rightarrow Q)] \equiv (P \vee Q) \wedge \neg P \wedge \neg Q$

2. 重复进行推理

1. $P \vee Q$ [¬ Conclusion]
2. $\neg P$ [¬ Conclusion]
3. $\neg Q$ [¬ Conclusion]
4. Q [1, 2. Resolution]

3. 得到矛盾，得证

5. $\square$ [3, 4. Resolution]

性质讨论： 可靠性与完备性 **Soundness and Completeness**

- An inference procedure (and hence a logic) is sound if and only if it preserves truth
- In other words $\vdash$ is sound iff whenever $\lambda \vdash \rho$, then $\lambda \models \rho$
- A logic is complete if and only if it is capable of proving all truths
- In other words, whenever $\lambda \models \rho$, then $\lambda \vdash \rho$

可靠性： 如果p可以推理得到q, 那么p语义蕴含q

完备性： 如果p语义蕴含q, 那么p可以推理得到q

命题逻辑的推理

这些性质到底是什么意思?

Syntax: 语法，相当于命题构成和推理规则 → p可以推理得到q

Semantic: 语义，命题蕴含的意义，即真值 → p为真时，q也为真

要分清楚几个符号:

semantically ($\models$)	语义，前键是一个集合，后键是一个命题	$p \to q,\ \neg p \to \neg q \models \neg p \leftrightarrow \neg q$
syntactically ($\vdash$)	语法，前键是一个集合，后键是一个命题	$(G \vee H) \to (\neg J \wedge \neg K),\ G \vdash \neg J$

$\to$ 蕴含：命题逻辑的二元运算符 $p \to q$

命题逻辑的推理

Decidability 可判定性

- A logic is decidable if and only if there is a mechanical procedure that, when asked $\lambda \vdash \rho$, can eventually halt and answer "yes" or halt and answer "no"
- Propositional logic is decidable

逻辑体系对于任何命题都可以最终得到确切的二元回答（对还是错）

命题逻辑满足可判定性

一阶逻辑 First Order Logic (FOL)

常量 Constant Symbols: a, b, ..., Mary (objects)

变量 Variables: x, y, ...

函数 Function Symbols: f, mother of, sine, ...

谓词 Predicate Symbols: Mother, likes, ...

量词 Quantifiers: ∀ (universal); ∃ (existential)

一阶逻辑 First Order Logic (FOL)

一阶逻辑中一个(原子)命题:

Eg. 素数(2), 表示命题“2是个素数”。

Eg. 好朋友(张三, 李四), 表示命题“张三和李四是好朋友”。

Everyone likes lying on the beach : $\forall x$ Beach(x)

Someone likes Fido : $\exists x$ Likes(x, Fido)

No one likes Fido : $\neg\exists x$ Likes(x, Fido)

Fido doesn’t like everyone : $\neg\forall x$ Likes(Fido, x)

All cats are mammals : $\forall x$ (Cat(x) $\rightarrow$ Mammal(x))

注意区分谓词和函数

一阶逻辑 First Order Logic (FOL)

把命题转化为FOL
Example

Translate the following sentences into first-order logic:

1. "All employees have income."
2. "Some employees are on holidays."
3. "No employees are unemployed."
4. "Some employees are not satisfied by their salary policy."

1 $\forall x[employee(x) \rightarrow has\_income(x)]$.

2 $\exists x[employee(x) \wedge on\_holidays(x)]$.

3 $\forall x[employee(x) \rightarrow \neg unemployed(x)]$.

4 $\exists x[employee(x) \wedge \neg satisfied(x, salary\_policy)]$,

where '$salary\_policy$' is a constant denoting a particular salary policy.

一阶逻辑 First Order Logic (FOL)

把命题转化为FOL
More example

Translate the following sentences into first-order logic:

1. "I always travel somewhere not far from the city where I live."
2. "I sometimes travel somewhere not far from the city where I live."
3. "I sometimes visit all my favourite places not far from the city where I live."

- $travel(t, p)$ – "at time t I travel to place p";
- $far\_from(p_1, p_2)$ – "places p_1, p_2 are far from each other";
- $favorite(p)$ – "p is one of my favourite places".

1 $\forall t \exists p \left[travel(t, p) \wedge \neg far\_from(p, my\_city) \right]$,

where 'my_city' is a constant denoting "my city".

2 $\exists t \exists p \left[travel(t, p) \wedge \neg far\_from(p, my\_city) \right]$.

3 $\exists t \forall p \left[(favorite(p) \wedge \neg far\_from(p, my\_city)) \rightarrow travel(t, p) \right]$.

一阶逻辑 First Order Logic (FOL)

容易confusing的一个问题

对全称量词, 把限定谓词作为蕴含式之前件加入, 即 $\forall x(P(x)\rightarrow\ldots)$。

所有的鸟都会飞: $\forall x(Bird(x) \rightarrow Fly(x))$

对存在量词, 把限定量词作为一个合取项加入, 即 $\exists x(P(x)\wedge\ldots)$。

有的鸟会飞: $\exists x(Bird(x) \wedge Fly(x))$

一阶逻辑 First Order Logic (FOL)

如何表达“不存在最大的整数”？

阐述过程 option1：
1. 如何表示整数：G(x) = x是整数
2. 如何表示大小关系：D(x, y) = x大于y
3. 如何表示最大：一个数比任何数都大
4. 先考虑存在，然后取反即可

$$\neg\exists x(G(x)\Lambda\forall y(G(y)\rightarrow D(x,y))$$

阐述过程 option2：
1. 如何表示整数：G(x) = x是整数
2. 如何表示大小关系：D(x, y) = x大于y
3. 如何表示没有最大：任何数都能找个一个数比其大

$$\forall x(G(x)\rightarrow\exists y(G(y)\wedge D(y,x))$$

Tip：一般约定用大写英文字母作为谓词符号,用小写字母*f*,*g*,*h*等表示函数符号,用小写字母*x*,*y*,*z*等作为个体变量符号，用小写字母*a*,*b*,*c*等作为个体常量符号。

一个命题的表达方式不是唯一的

一阶逻辑 First Order Logic (FOL)

量词嵌套 Nested Quantifiers

Everything likes everything — ∀x ∀y Likes(x, y)

Everything likes something — ∀x ∃y Likes(x, y)

There is something liked by everything — ∃y ∀x Likes(x, y)

对于所有的自然数, 均有x+y>x

$$\forall x \forall y (N(x) \wedge N(y) \rightarrow S(x, y, x))$$

某些人对某些食物过敏

$$\exists x \exists y (M(x) \wedge F(y) \wedge G(x, y))$$

一阶逻辑 First Order Logic (FOL)

量词的作用域 Scope of Quantifiers

就近原则：量词的作用域局限于其成为主要逻辑符号的范围

$Q(x) \rightarrow \forall y\, P(x,\, y)$ — scope of $\forall y$ is $P(x,\, y)$

$\forall z\, P(z) \rightarrow \neg Q(z)$ — scope of $\forall z$ is $P(z)$ but not $Q(z)$

$\exists x(P(x) \rightarrow \forall x\, P(x))$

$\forall x(P(x) \rightarrow Q(x)) \rightarrow (\forall x\, P(x) \rightarrow \forall x\, Q(x))$

Conjunctive Normal Form(CNF)

以合取式表达的命题，如 P∧Q ∧M

1. Eliminate implication

$$\phi \rightarrow \psi \equiv \neg\phi \vee \psi$$

2. Move negation inwards (negation normal form)

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$$
$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$
$$\neg\, \forall x\, \phi \equiv \exists x\, \neg\phi$$
$$\neg\, \exists x\, \phi \equiv \forall x\, \neg\phi$$
$$\neg\neg\phi \equiv \phi$$

3. Standardise variables

$(\forall x\, P(x)) \vee (\exists x\, Q(x))$

becomes $(\forall x\, P(x)) \vee (\exists y\, Q(y))$

Conjunctive Normal Form(CNF)

以合取式表达的命题，如 P∧Q ∧M

4. Skolemise

$$\exists x\, P(x) \Rightarrow P(a)$$
$$\forall x \exists y\, P(x,\, y) \Rightarrow \forall x\, P(x,\, f(x))$$
$$\forall x \forall y \exists z\, P(x,\, y,\, z) \Rightarrow \forall x \forall y\, P(x,\, y,\, f(x,\, y))$$

5. Drop universal quantifiers

6. Distribute $\wedge$ over $\vee$

$$(\phi \wedge \psi) \vee \chi \equiv (\phi \vee \chi) \wedge (\psi \vee \chi)$$

7. Flatten nested conjunctions and disjunctions

$$(\phi \wedge \psi) \wedge \chi \equiv \phi \wedge \psi \wedge \chi; (\phi \vee \psi) \vee \chi \equiv \phi \vee \psi \vee \chi$$

Skolemise: 消去存在量词时，同时还要进行变量替换。变量替换分两种情况：

①若该存在量词在某些全称量词的辖域内，则用这些全称量词指导变量的一个函数代替该存在量词辖域中的相应约束变量，这样的函数称为Skolem函数；

②若该存在量词不在任何全称量词的辖域内，则用一个常量符号代替该存在量词辖域中的相应约束变量，这样的常量符号称为Skolem常量。

Conjunctive Normal Form(CNF)

以合取式表达的命题，如 P∧Q ∧M

$$\neg\exists x\forall y\forall z((P(y)\vee Q(z))\rightarrow(P(x)\vee Q(x)))$$

$$\neg\exists x\forall y\forall z(\neg(P(y)\vee Q(z))\vee(P(x)\vee Q(x))) \text{ [Eliminate } \rightarrow\text{]}$$

$$\forall x\neg\forall y\forall z(\neg(P(y)\vee Q(z))\vee(P(x)\vee Q(x))) \text{ [Move } \neg \text{ inwards]}$$

$$\forall x\exists y\neg\forall z(\neg(P(y)\vee Q(z))\vee(P(x)\vee Q(x))) \text{ [Move } \neg \text{ inwards]}$$

$$\forall x\exists y\exists z\neg(\neg(P(y)\vee Q(z))\vee(P(x)\vee Q(x))) \text{ [Move } \neg \text{ inwards]}$$

$$\forall x\exists y\exists z(\neg\neg(P(y)\vee Q(z))\wedge\neg(P(x)\vee Q(x))) \text{ [Move } \neg \text{ inwards]}$$

$$\forall x\exists y\exists z((P(y)\vee Q(z))\wedge(\neg P(x)\wedge\neg Q(x))) \text{ [Move } \neg \text{ inwards]}$$

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

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

Skolemise

$\exists x\, P(x) \Rightarrow P(a)$

$\forall x\exists y\, P(x,\ y) \Rightarrow \forall x\, P(x,\ f(x))$

$\forall x\forall y\exists z\, P(x,\ y,\ z) \Rightarrow \forall x\forall y\, P(x,\ y,\ f(x,\ y))$

Conjunctive Normal Form(CNF)

Unification

Unification takes two atomic formulae and returns a substitution that makes them look the same

Example:

$$\{x/a,\ y/z,\ w/f(b,\ c)\}$$

使用 / 后的变量替换前者，注意每个符号只能替换一次，不允许嵌套替换或者重复替换

替换变量在推理时十分有用

Conjunctive Normal Form(CNF)

Skolemization Example

Exp 1.

- There exists a philosopher with students.
  $\exists x \exists y[Philo(x) \wedge StudentOf(y, x)]$
- Skolemize: substitute x by a and y by b
  $Philo(a) \wedge StudentOf(b, a)$

Exp 2.

- Every philosopher writes at least one book.
  $\forall x[Philo(x) \rightarrow \exists y[Book(y) \wedge Write(x, y)]]$
- Eliminate Implication:
  $\forall x[\neg Philo(x) \vee \exists y[Book(y) \wedge Write(x, y)]]$
- Skolemize: substitute y by $g(x)$
  $\forall x[\neg Philo(x) \vee [Book(g(x)) \wedge Write(x, g(x))]]$

Exp 3.

- All students of a philosopher read one of their teacher's books.
  $\forall x \forall y[Philo(x) \wedge StudentOf(y, x) \rightarrow$
  $\exists z[Book(z) \wedge Write(x, z) \wedge Read(y, z)]]$
- Eliminate Implication:
  $\forall x \forall y[\neg Philo(x) \vee \neg StudentOf(y, x) \vee \exists z[Book(z) \wedge$
  $Write(x, z) \wedge Read(y, z)]]$
- Skolemize: substitute z by $h(x, y)$
  $\forall x \forall y[\neg Philo(x) \vee \neg StudentOf(y, x) \vee [Book(h(x, y)) \wedge$
  $Write(x, h(x, y)) \wedge Read(y, h(x, y))]]$

First-Order Resolution 一阶逻辑的推理

$$\vdash \exists x(P(x) \rightarrow \forall x P(x))$$

转化为CNF $\mathrm{CNF}(\neg\exists x(P(x) \rightarrow \forall x P(x)))$

$\forall x \neg(\neg P(x) \vee \forall x\, P(x))$ [Drive ¬ inwards]

$\forall x(\neg\neg P(x) \wedge \neg\forall x\, P(x))$ [Drive ¬ inwards]

$\forall x(P(x) \wedge \exists x\, \neg P(x))$ [Drive ¬ inwards]

$\forall x(P(x) \wedge \exists z\, \neg P(z))$ [Standardise Variables]

$\forall x(P(x) \wedge \neg P(f(x)))$ [Skolemise]

$P(x) \wedge \neg P(f(x))$ [Drop $\forall$]

1. $P(x)$ [¬ Conclusion]
2. $\neg P(f(y))$ [¬ Conclusion]
3. $P(f(y))$ [1. $\{x/f(y)\}$]
4. $\square$ [2, 3. Resolution]

First-Order Resolution 一阶逻辑的推理

$$\vdash \neg\exists x \left(P(f(x)) \vee Q\big(g(x)\big)\right) \vee P(y) \vee Q(z)$$

$$CNF\ \neg(\neg\exists x \left(P\big(f(x)\big) \vee Q\big(g(x)\big)\right) \vee P(y) \vee Q(z))$$

$$\exists x \left(P\big(f(x)\big) \vee Q\big(g(x)\big)\right) \wedge \neg P(y) \wedge \neg Q(z))$$

1. $P\big(f(a)\big) \vee Q\big(g(a)\big)$
2. $\neg P(y)$
3. $\neg Q(z)$
4. $\neg P(f(a))$ [2. $\{y/f(a)\}$]
5. $\neg Q(g(a))$ [3. $\{z/g(a)\}$]
6. $Q(g(a))$ [4,1. Resolution]
7. ■ [6,7. Resolution]

First-Order Resolution 一阶逻辑的推理

Exercise 1

"The barber is a man in town who shaves all those, and only those, men in town who do not shave themselves."

Translate the claim into FOL and check if its satisfiability

First-Order Resolution 一阶逻辑的推理

Exercise 1. Solution

$$\forall x[shaves(barber, x) \leftrightarrow \neg shaves(x, x)].$$

CNF

$$\begin{aligned}
& \forall x[shaves(barber, x) \leftrightarrow \neg shaves(x, x)] \\
\leftrightarrow \quad & shaves(barber, x) \leftrightarrow \neg shaves(x, x) \\
\leftrightarrow \quad & \neg shaves(barber, x) \vee \neg shaves(x, x) \wedge \\
& shaves(barber, x) \vee \neg\neg shaves(x, x) \\
\leftrightarrow \quad & \neg shaves(barber, x) \vee \neg shaves(x, x) \wedge \\
& shaves(barber, x) \vee shaves(x, x).
\end{aligned}$$

Resolution

1. $\neg shaves(barber, x_1) \vee \neg shaves(x_1, x_1)$
2. $shaves(barber, x_2) \vee shaves(x_2, x_2)$
3. $\neg shaves(barber, barber)$
   – (fctr): 1 with $x_1 = barber$
4. $shaves(barber, barber)$
   – (fctr): 2 with $x_2 = barber$
5. $\emptyset$ – (res): 3, 4.

First-Order Resolution 一阶逻辑的推理

Exercise 2.

Prove the following formula using resolution, where *b* is a constant:

$$\Big[\forall y Q(b,y) \wedge \forall x \forall y [Q(x,y) \rightarrow Q(s(x),s(y))]\Big] \rightarrow \exists z [Q(b,z) \wedge Q(z,s(s(b)))].$$

First-Order Resolution 一阶逻辑的推理

Exercise 2. Solution

CNF

$$
\begin{aligned}
& \neg\Big\{\Big[\forall y Q(b,y) \wedge \forall x \forall y [Q(x,y) \rightarrow Q(s(x),s(y))]\Big] \\
\rightarrow \quad & \exists z [Q(b,z) \wedge Q(z,s(s(b)))]\Big\} \\
\leftrightarrow \quad & \Big[\forall y Q(b,y) \wedge \forall x \forall y [Q(x,y) \rightarrow Q(s(x),s(y))]\Big] \wedge \\
& \neg\exists z [Q(b,z) \wedge Q(z,s(s(b)))] \\
\leftrightarrow \quad & \Big[\forall y Q(b,y) \wedge \forall x \forall y [Q(x,y) \rightarrow Q(s(x),s(y))]\Big] \wedge \\
& \forall z \neg [Q(b,z) \wedge Q(z,s(s(b)))].
\end{aligned}
$$

Resolution

This gives the following clauses (after renaming variables):

1. $Q(b, y_1)$;
2. $\neg Q(x_2, y_2) \vee Q(s(x_2), s(y_2))$;
3. $\neg Q(b, z_3) \vee \neg Q(z_3, s(s(b)))$.

Proof by resolution:

4. $\neg Q(z_4, s(s(b)))$ – (res): 1, 3 with $y_1 = z_3$ and renaming z_3/z_4
5. $\emptyset$ – (res): 1, 4 with $z_4 = b, y_1 = s(s(b))$.

First-Order Resolution 一阶逻辑的推理

	命题逻辑	一阶逻辑
Soundness 可靠性	Yes	Yes
Completeness 完备性	Yes	Yes
Decidability 可判性	Yes	No

Default Logic 缺省逻辑

缺省逻辑可以表达像“缺省的，某个事物是真的”的事实；
相反的，标准逻辑只能表达某个事物为真或某个事物为假。

$$\frac{Prerequisite : Justification_1, \ldots, Justification_n}{Conclusion}$$

Exp. “鸟通常会飞”

这个规则可以在标准逻辑中表达为要么“所有鸟都会飞”，这与企鹅不会飞的事实相矛盾；要么“除了企鹅、鸵鸟...的所有鸟都会飞”，这要求规则指定出所有的例外。缺省逻辑致力于形式化像这样的推理规则，而不需要明确提及所有的例外。

$$D = \left\{ \frac{Bird(X) : Flies(X)}{Flies(X)} \right\}$$

Default Logic 缺省逻辑

$$D = \left\{ \frac{Bird(X) : Flies(X)}{Flies(X)} \right\}$$

W = { Bird（秃鹫），Bird（企鹅），¬ Flies（企鹅），Flies（鹰）}。

1. 依据这个缺省规则，秃鹫会飞，因为前件Bird（秃鹫）是真的，并且论据Flies（秃鹫）与当前已知不矛盾。

2. 相反的，Bird（企鹅）不允许得出结论Flies（企鹅）:即使缺省的前件Bird（企鹅）是真的，论据Flies（企鹅）与已知相矛盾。

3. 不能得出结论Bird（鹰），因为缺省规则只允许从Bird（X）推出Flies (X)，但不能反过来。

Nonmonotonic Reasoning

经典逻辑当中，条件越多则得到的结论越多，这称为单调性monotonicity
但实际情况往往并非如此，新获得的条件可能与之前的条件冲突，抵消我们已知的

- It is usually not possible to write down all we would like to say about a domain
- Inferences in classical logic simply make implicit knowledge explicit; we would also like to reason with tentative statements
- Sometimes we would like to represent knowledge about something that is not *entirely* true or false; uncertain knowledge

The Closed World Assumption(CWA)

如果我们没有证据证明一个命题为真，则默认它为假

Knowledge Representation

Vocabulary 定义方式和命题逻辑基本一样

named individuals

john, countryTown, faultyInsuranceCorp, fic, johnQsmith, ...

basic types

Person, Place, Man, Woman, ...

attributes

Rich, Beautiful, Unscrupulous, ...

relationships

LivesAt, MarriedTo, DaughterOf, HairDresserOf, HadAnAffairWith, Blackmails, ...

functions

fatherOf, ceoOf, bestFriendOf, ...

Knowledge Representation

Basic Facts

type facts

Man(john),

Woman(jane),

Company(faultyInsuranceCorp)

property facts

Rich(john),

¬HappilyMarried(jim),

WorksFor(jim,fic)

equality facts

john = ceoOf(fic),

fic = faultyInsuranceCorp,

bestFriendOf(jim) = john

Complex Facts

Universal abbreviations

$\forall y[\text{Woman}(y) \wedge y \neq \text{jane} \supset \text{Loves}(y,\text{john})]$

$\forall y[\text{Rich}(y) \wedge \text{Man}(y) \supset \text{Loves}(y,\text{jane})]$

$\forall x \forall y[\text{Loves}(x,y) \supset \neg\text{Blackmails}(x,y)]$

possible to express without quantifiers

Incomplete knowledge

$\text{Loves}(\text{jane},\text{john}) \vee \text{Loves}(\text{jane},\text{jim})$

which?

$\exists x[\text{Adult}(x) \wedge \text{Blackmails}(x,\text{john})]$

who?

cannot write down more complete version

Closure axioms

$\forall x[\text{Person}(x) \supset x=\text{jane} \vee x=\text{john} \vee x=\text{jim} \ldots]$

$\forall x \forall y[\text{MarriedTo}(x,y) \supset \ldots]$

$\forall x[\, x=\text{fic} \vee x=\text{jane} \vee x=\text{john} \vee x=\text{jim} \ldots]$

limits domain of discourse

also useful to have jane ≠ john ...

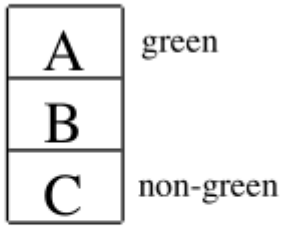
Knowledge Representation

Example:

Three blocks stacked.

Top one is green.

Bottom one is not green.



Is there a green block directly on top of a non-green block?

Knowledge Representation

$S = \{\text{On(a,b)}, \text{On(b,c)}, \text{Green(a)}, \neg\text{Green(c)}\}$

all that is required

$\alpha = \exists x \exists y[\text{Green}(x) \wedge \neg\text{Green}(y) \wedge \text{On}(x,y)]$

Claim: $S \models \alpha$

还有其他的推理方式吗?

Proof:

Let $\boldsymbol{I}$ be any interpretation such that $\boldsymbol{I} \models S$.

Case 1: $\boldsymbol{I} \models \text{Green(b)}$.

$\therefore \boldsymbol{I} \models \text{Green(b)} \wedge \neg\text{Green(c)} \wedge \text{On(b,c)}$.

$\therefore \boldsymbol{I} \models \alpha$

Case 2: $\boldsymbol{I} \not\models \text{Green(b)}$.

$\therefore \boldsymbol{I} \models \neg\text{Green(b)}$

$\therefore \boldsymbol{I} \models \text{Green(a)} \wedge \neg\text{Green(b)} \wedge \text{On(a,b)}$.

$\therefore \boldsymbol{I} \models \alpha$

Either way, for any $\boldsymbol{I}$, if $\boldsymbol{I} \models S$ then $\boldsymbol{I} \models \alpha$.

So $S \models \alpha$. QED

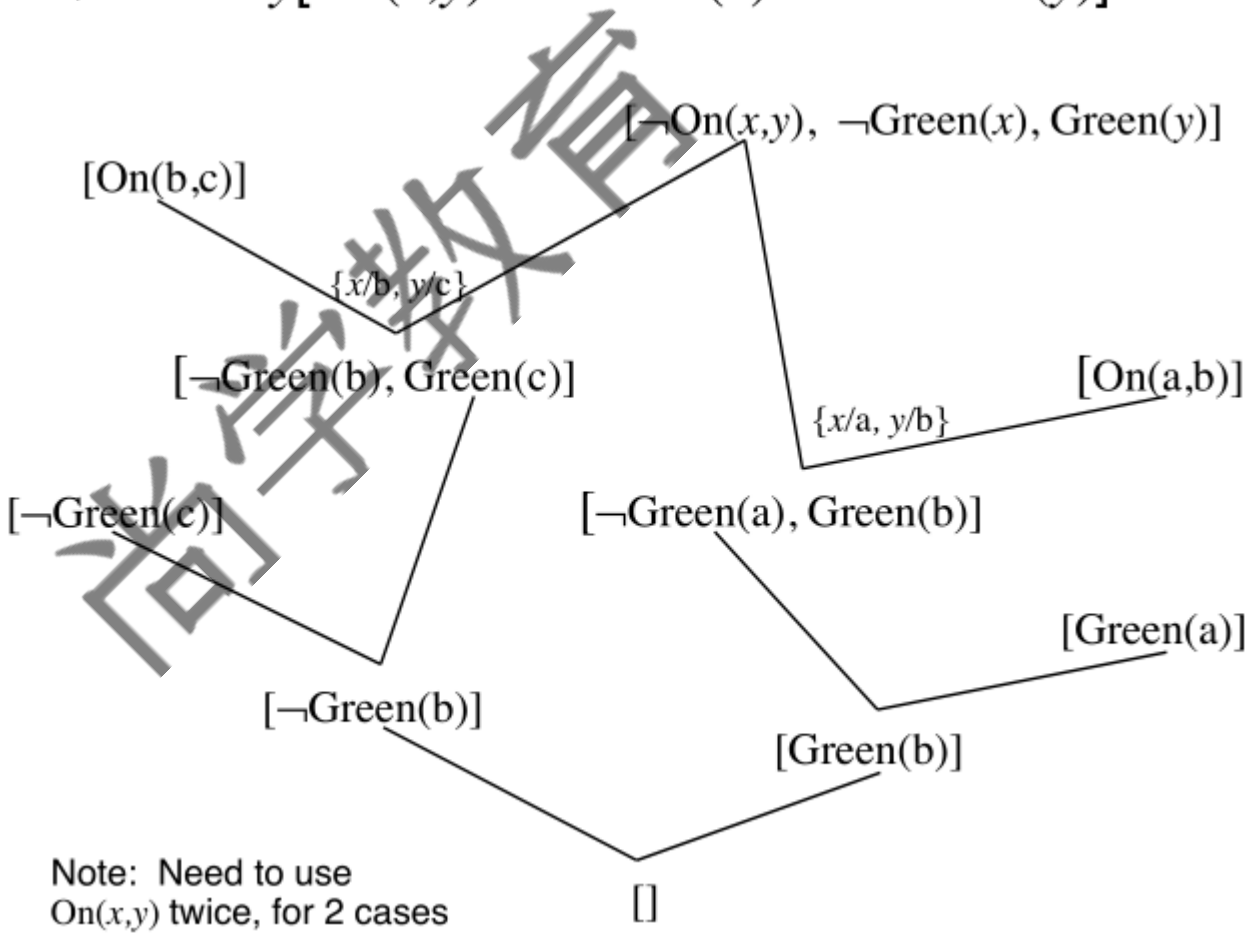
Knowledge Representation

使用Refutation system

$$KB = \{On(a,b),\ On(b,c),\ Green(a),\ \neg Green(c)\}$$

already in CNF

$$Q = \exists x \exists y [On(x,y) \wedge Green(x) \wedge \neg Green(y)]$$



Note: Need to use On(x,y) twice, for 2 cases

Knowledge Representation

Is there a company whose CEO loves Jane?

$$\exists x\ [\text{Company}(x) \wedge \text{Loves}(\text{ceoOf}(x), \text{jane})]\ \ ??$$

Suppose $\boldsymbol{I} \models$ KB.

Then $\boldsymbol{I} \models \text{Rich}(\text{john}),\ \text{Man}(\text{john}),$

and $\boldsymbol{I} \models \forall y[\text{Rich}(y) \wedge \text{Man}(y) \supset \text{Loves}(y, \text{jane})]$

so $\boldsymbol{I} \models \text{Loves}(\text{john}, \text{jane}).$

Also $\boldsymbol{I} \models \text{john} = \text{ceoOf}(\text{fic}),$

so $\boldsymbol{I} \models \text{Loves}(\text{ceoOf}(\text{fic}), \text{jane}).$

Finally $\boldsymbol{I} \models \text{Company}(\text{faultyInsuranceCorp}),$

and $\boldsymbol{I} \models \text{fic} = \text{faultyInsuranceCorp},$

so $\boldsymbol{I} \models \text{Company}(\text{fic}).$

Thus, $\boldsymbol{I} \models \text{Company}(\text{fic}) \wedge \text{Loves}(\text{ceoOf}(\text{fic}), \text{jane}),$

and so

$$\boldsymbol{I} \models \exists x\ [\text{Company}(x) \wedge \text{Loves}(\text{ceoOf}(x), \text{jane})].$$

KB={
John是一个富有的男士，
所有富有的男士都喜欢jane，
John是fic的CEO，
fic是一家名为faultyInsuranceCorp的公司
}

Knowledge Representation

Horn clauses & SLD resolution

Horn clause = at most one +ve literal in clause 带有最多一个肯定项的子句

- positive / definite clause = exactly one +ve literal

  $[\neg p_1, \neg p_2, ..., \neg p_n, q]$

- negative clause = no +ve literals

  $[\neg p_1, \neg p_2, ..., \neg p_n]$

Note

$[\neg p_1, \neg p_2, ..., \neg p_n, q]$ is a representation for

$(\neg p_1 \vee \neg p_2 \vee ... \vee \neg p_n \vee q)$ or

$[(p_1 \wedge p_2 \wedge ... \wedge p_n) \supset q]$

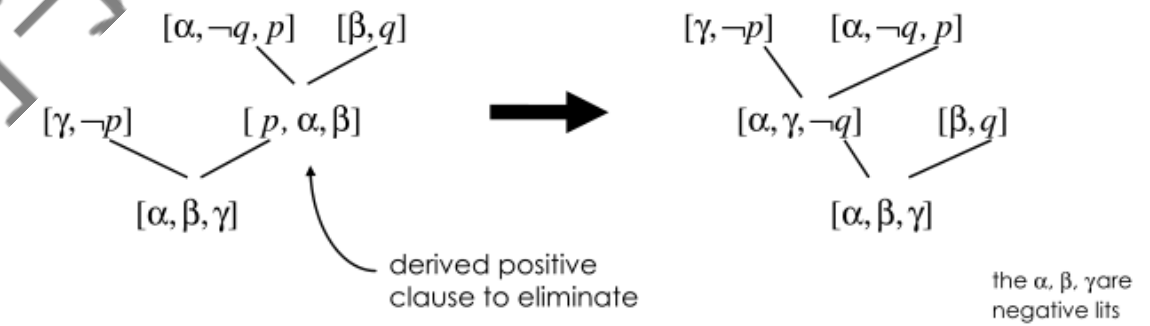
Knowledge Representation

Horn clauses & SLD resolution

和命题逻辑的推理思路一样，
不断抵消矛盾项

Only two possibilities:

It is possible to rearrange derivations (of negative clauses) so that all new derived clauses are negative clauses



Knowledge Representation

Horn clauses & SLD resolution

KB:

FirstGrade

FirstGrade $\Rightarrow$ Child

Child $\wedge$ Male $\Rightarrow$ Boy

Kindergarten $\Rightarrow$ Child

Child $\wedge$ Female $\Rightarrow$ Girl

Female

1	FirstGrade
2	¬FirstGrade ∨ Child
3	¬Child ∨ ¬Male ∨ Boy
4	¬Kindergarten ∨ Child
5	¬Child ∨ ¬Female ∨ Girl
6	Female

Show KB $\cup$ {~Girl} unsatisfiable

[¬Girl]

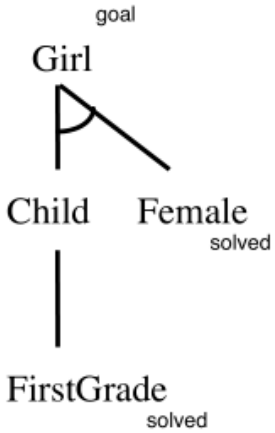
[¬Child, ¬Female]

[¬Child]

[¬FirstGrade]

[]

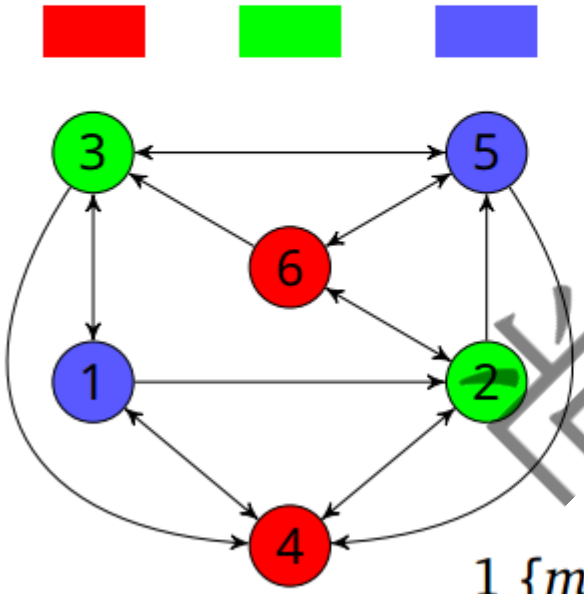
or



A goal tree whose nodes are atoms, whose root is the atom to prove, and whose leaves are in the KB

Definition: graph colouring problem

Input: graph with vertices V and edges $E \subseteq V \times V$, set of colors C.
Is there a mapping $m : V \to C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?



$c(r).\ c(g).\ c(b).$
$v(1).\ \ldots\ v(6).$
$e(1,2).\ e(1,3).\ e(1,4).$
$e(2,4).\ e(2,5).\ e(2,6).$
$e(3,1).\ e(3,4).\ e(3,5).$
$e(4,1).\ e(4,2).$
$e(5,3).\ e(5,4).\ e(5,6).$
$e(6,2).\ e(6,3).\ e(6,5).$

$1\ \{m(X,C) : c(C)\}\ 1 \text{ :- } v(X).$ guess mapping m
$\text{:- } e(X,Y), m(X,C), m(Y,C).$ verify $m(X) \neq m(Y)$

基本语法

ASP中语句分为两种：事实+逻辑语句

Fact:

eagle(eddy).
penguin(tux).
e(1,2). v(5).
color(vertex).

Logic:

bird(X) :- penguin(X).
:- edge(X,Y), color(X,C), color(Y,C).
l {a, b, c, d} k
vertex_color(v1, Y)： color(Y)

1. Basic Logic

符号 :- 连接的语句，表示如果右边式子成立则左边式子成立。

2. Integrity Constraints

所有在符号':-'左边什么都没有的规则就是约束规则，它表示右边的条件不能同时被全部满足。

3. Conditions

符号 ： 连接的两个语句，如vertex_color(v1, Y) : color(Y)，表示前者里面的元素必须满足后面的语句。条件可以连续使用，如queen(I, J) : row(I) : col(J).

4. Choice Rules

l {a, b, c, d} k 表示大括号里的元素最少有l个，且最多有k个成立。

Interpretation

Definition: normal logic program (NLP)

A **normal logic program** P is a set of (normal) rules of the form

$$A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n.$$

where A, B_i, C_j are atomic propositions.
When $m = n = 0$, we omit the "$\leftarrow$" and just write A.

For such a rule r, we define:

- $\text{Head}(r) = \{A\}$
- $\text{Body}(r) = \{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$

In code, r is written as A `:-` $B_1, \ldots, B_m$, `not` $C_1, \ldots,$ `not` C_n.

Definition: interpretation, satisfaction

A **interpretation** S is a set of atomic propositions.

S **satisfies** $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff
$A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

- S satisfies rule iff S satisfies the head or falsifies the body
- S falsifies body iff S falsifies some B_i or satisfies some C_j

Interpretation

Remember: 要么满足head, 要么不满足body

Ex.: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not } b.\}$

$S = \{a, b, c\}$ satisfies a, but it does not satisfy $(\text{not } b)$.

It satisfies $c \leftarrow a, b$ because it satisfies the head because $c \in S$

It satisfies $d \leftarrow a, \text{not } b$ because it falsifies the body because $b \in S$

Definition: stable model for programs without negation

For P without negated literals:
S is a **stable model** of P iff
S is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

Ex.: $P = \{a. \quad c \leftarrow a, b.\}$

$S_1 = \{a\}$ is a stable model of P
$S_2 = \{a, b\}$ is not a stable model of P
$S_3 = \{a, b, c\}$ is not a stable model of P

Theorem: unique-model property

If P is negation-free (i.e., contains no $(\text{not}\, C)$), then there is exactly one stable model, which can be computed in linear time.

stable model without negation

在没有否定形式条件情况下，stable model非常容易计算（线性复杂度），而且只有一个

Theorem: unique-model property

If P is negation-free (i.e., contains no $(\text{not}\, C)$), then there is exactly one stable model, which can be computed in linear time.

Compute stable model of a negation-free P by *unit propagation*:

- $S^0 = \{\}$
- $S^{i+1} = S^i \cup \bigcup_{r \in P: S \text{ satisfies Body}(r)} \text{Head}(r)$ until $S^{i+1} = S^i$

Ex.: $P_1 = \{a. \quad b \leftarrow a.\}$

$S^0 = \{\}$ $S^1 = \{a\}$ $S^2 = \{a, b\}$ Fixpoint

Ex.: $P_2 = \{a \leftarrow b. \quad b \leftarrow a.\}$

$S^0 = \{\}$ Fixpoint

Ex.: $P_3 = \{a \leftarrow b. \quad b \leftarrow a. \quad a.\}$

$S^0 = \{\}$ $S^1 = \{a\}$ $S^2 = \{a, b\}$ Fixpoint

stable model with negation

Definition: reduct

The **reduct** P^S of P relative to S is the least set such that
if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule r from P,

- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to P^S

如果有否性形式的条件，则考虑：
1. 如果否定式和P^S中的元素冲突，则把所有带有该否定式的条件整个扔掉
2. 反之只把条件中这个否定的部分去掉，其余的加入到P^S中

Ex.: $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \sout{\text{not } b}.\}$

$S_2 = \{a, b\} \quad \Rightarrow \quad P^{S_2} = \{a. \quad c \leftarrow a, b. \quad \sout{d \leftarrow a, \text{not } b.}\}$

$S_3 = \{a, d\} \quad \Rightarrow \quad P^{S_3} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \sout{\text{not } b}.\}$

stable model with negation

Definition: stable model for programs with negation

For P with negated literals:
S is a **stable model** of P iff S is a stable model of P^S.

现在我们只需要考察S是否是P^S的stable model就可以了， P^S中不含否定形式

Ex.: $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\}$ ✗

$S_2 = \{a, b\} \quad \Rightarrow \quad P^{S_2} = \{a. \quad c \leftarrow a, b.\}$ ✗

$S_3 = \{a, d\} \quad \Rightarrow \quad P^{S_3} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\}$ ✓

stable model with negation

Definition: stable model for programs with negation

For P with negated literals:
S is a **stable model** of P iff S is a stable model of P^S.

现在我们只需要考察S是否是P^S的stable model就可以了， P^S中不含否定形式

Ex.: $P = \{a \leftarrow \text{not}\, b. \quad b \leftarrow \text{not}\, a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad b\}$ ✗
$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a.\}$ ✓
$S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b.\}$ ✓
$S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{\}$ ✗

Two stable models!

Ex.: $P = \{a \leftarrow \text{not}\, a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a.\}$ ✗
$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{\}$ ✗

No stable model!

Entailment & cautious monotonicity

Definition: entailment, cautious monotonicity

P **entails** a rule r iff every stable model of P satisfies r.

P is **cautiously monotonic** iff

for all rules r_1, r_2, if P entails r_1 and r_2, then $P \cup \{r_1\}$ entails r_2.

If P is cautiously monotonic, a solver can iteratively augment it with already proved lemmas. *Bad news*: it does **not** hold in general.

Ex.: $P = \{a \leftarrow \text{not } b. \quad b \leftarrow c, \text{not } a. \quad c \leftarrow a.\}$

$S_1 = \{a, c\} \Rightarrow P^{S_1} = \{a. \quad c \leftarrow a.\}$ ✓

(no other stable model S: $b \notin S \Rightarrow a \in S \Rightarrow c \in S$ and $b \in S \Rightarrow c \in S \Rightarrow a \in S \Rightarrow b \notin S$)

$S_1 = \{a, c\} \Rightarrow (P \cup \{c.\})^{S_1} = \{a. \quad c \leftarrow a. \quad c.\}$ ✓

$S_2 = \{b, c\} \Rightarrow (P \cup \{c.\})^{S_2} = \{b \leftarrow c. \quad c \leftarrow a. \quad c.\}$ ✓

(no other stable model S: $c \in S$ and $a \notin S \Rightarrow b \in S$ and $b \notin S \Rightarrow a \in S$)

P entails $c.$ and $a.$ But $P \cup \{c.\}$ does not entail a.

Satisfiability

Definition: DNF

A formula ψ is in Disjunctive Normal Form iff
ψ is of the form $(d_1 \vee \ldots \vee d_k)$, where each d_i is of the form $(x_{i1} \wedge \ldots \wedge x_{il_i})$, where each x_{ij} is a literal.

Ex.: $(p \wedge q) \vee (\neg p \wedge \neg q) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$

Definition: CNF

A formula ϕ is in Conjunctive Normal Form iff
ϕ is of the form $(c_1 \wedge \ldots \wedge c_k)$, where each c_i is of the form $(x_{i1} \vee \ldots \vee x_{il_i})$, where each x_{ij} is a literal.

Ex.: $(\neg p \vee \neg q) \wedge (p \vee q) \wedge (\neg p \vee q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$

任何formula都可以转化为DNF或者CNF的形式

Satisfiability

布尔可满足性问题 Boolean satisfiability problem = SAT

Definition: SAT

Input: ϕ in CNF.
Problem: Is this formula satisfiable?

Why not DNF?

- Satisfiability for DNF is very easy, but ...
- DNF of formula may grow exponentially!

Why CNF?

- Structure of CNF is much simpler than of arbitrary formulas.
- Every formula ψ can be transformed to a formula ϕ such that:
  - ϕ is satisfiable iff ψ is satisfiable.
  - The size of ϕ is linear in the size of ψ.

DNF的复杂度可能指数型增大

CNF则为线性

Satisfiability

Syntax and Semantics Revisited先明确I和ϕ的定义

Syntax:

- A CNF formula ϕ is of the form $(c_1 \wedge \ldots \wedge c_k)$,
  where each c_i is of the form $(x_{i1} \vee \ldots \vee x_{il_i})$
  where each x_{ij} is a variable or a negated variable.
- We identify c_i with the set $\{x_{i1}, \ldots, x_{il_i}\}$.
- We identify ϕ with the set $\{c_1, \ldots, c_k\}$.
- We write $\bar{x}$ to flip the negation of a literal x: $\overline{\neg p} = p \quad \overline{p} = \neg p$

Semantics:

- A partial interpretation I is a consistent set of literals ($x \notin I$ or $\bar{x} \notin I$)
  It may falsify or satisfy a formula or neither (because it is partial)
- I satisfies a CNF formula ϕ iff I satisfies all clauses $c \in \phi$
  I falsifies a CNF formula ϕ iff I falsifies some clause $c \in \phi$
- I satisfies a clause c iff I satisfies some literal $x \in c$
  I falsifies a clause c iff I falsifies all literals $x \in c$
- I satisfies a literal x iff $x \in I$
  I falsifies a literal x iff $\bar{x} \in I$

Satisfiability

SAT Algorithm 1a: Nondeterministic

非确定性搜索解集

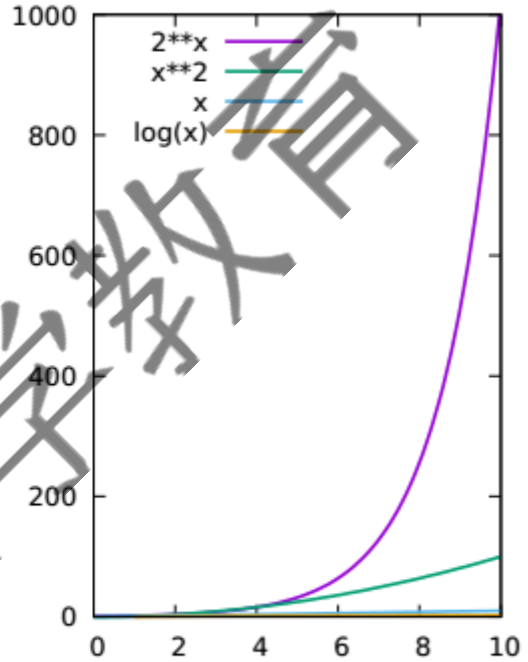
Let $I = \{\}$. Repeat:

1. If I falsifies some $c \in \phi$: return NO
2. Select a variable x such that $x, \bar{x} \notin I$
3. If there is none: return YES
4. Let $I = I \cup \{x\}$ or $I = I \cup \{\neg x\}$

存在明显问题：搜索过程耗时巨大，复杂度和变量个数成指数关系

Complexity Theory: Big O Notation

- Complexity is (usually) measured in the length n of the input:
  - $f(n) = \mathcal{O}(g(n))$ iff for some k, for large $n, f(n) \leq k \cdot g(n)$
  - Exponential complexity: $\mathcal{O}(k^n)$
  - Polynomial complexity: $\mathcal{O}(n^k)$
  - Linear complexity: $\mathcal{O}(n)$
  - Logarithmic complexity: $\mathcal{O}(\log n)$
  - Constant complexity: $\mathcal{O}(1)$



- Length of input = number of symbols:
  - Length of "$\neg(p \wedge q)$" $= 6$
  - Length of 173 in decimal $= 3$
  - Length of 173 in binary $= 8$

- Time complexity: number of computation steps.
- Space complexity: amount of memory used.
- Time is upper bound for space.

Satisfiability

Complexity Theory: P and NP

Definition: complexity class P, NP

$A \in$ P iff solvable in poly. time by a deterministic machine.
$A \in$ NP iff solvable in poly. time by a nondeterministic machine.

P问题可以在多项式时间之内求解

NP问题可以在多项式时间之内验证

Satisfiability

Complexity Theory: P and NP

除了P和NP问题之外：

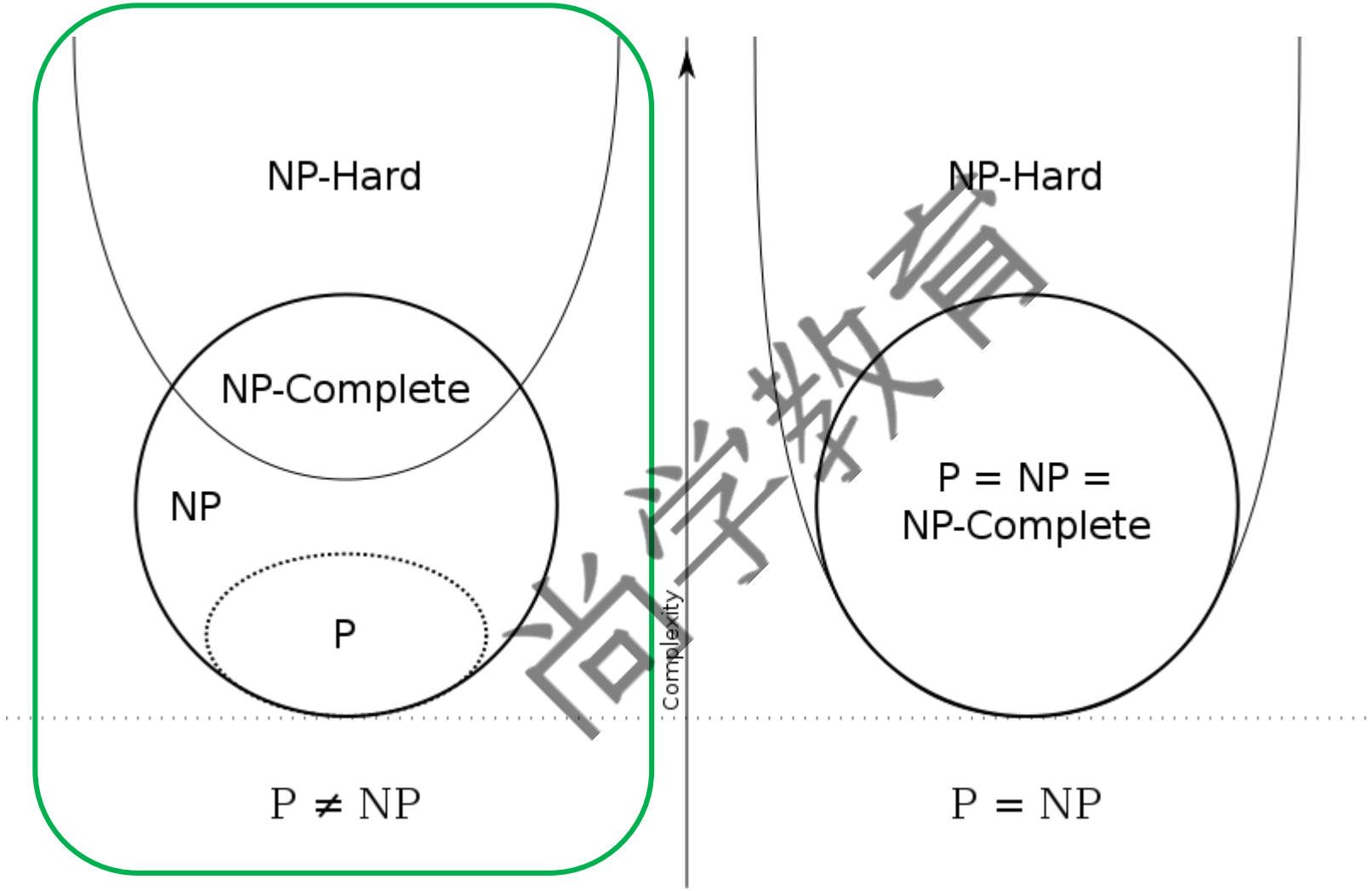
NP-hard问题：比所有NP问题都难的问题，可能没有多项式时间验证的方法。

NP-complete问题：

1. 首先是NP-Hard问题
2. 是NP问题

就是在多项式时间内可以验证一个candidate answer是不是真正的解，以及我们可以把任何一个NP问题在多项式的时间内转化为一个NP-complete问题（即**规约**）。

Complexity Theory: P and NP



Satisfiability

Complexity Theory: P and NP　　Reduction 规约

Definition: reduction from A to B

$A \leq_P B$ iff for some function f from A-instances to B-instances:
for all instances of x of A:
$f(x)$ is computable in polynomial time and
x is a "yes"-instance of A iff $f(x)$ is a "yes"-instance of B.

Definition: NP-hard, NP-complete

B is NP-hard　　iff $A \leq_P B$ for all $A \in$ NP.
B is NP-complete　iff $B \in$ NP and B is NP-hard.

Theorem: complexity of SAT

SAT and 3-SAT are NP-complete.

Satisfiability

Complexity Theory: P and NP

一些经典的NPC问题

有向哈密尔顿环
给定有向图H，判定其是否经过图中每个顶点且仅一次的回路。

无向哈密尔顿环
给定无向图G=(V,E)，判定其是否经过图中每个顶点且仅一次的回路。

0-1整数规划
给定一个整数矩阵C和一个整数向量d，判断是否存在一个0-1向量x，使得Cx=d

最小顶点覆盖问题
给定图 G=(V,E)和数k，判定是否存在包含大小至多为k的顶点覆盖。

集合覆盖问题
给定全集U，以及一个包含n个集合且这n个集合的并集为全集的集合S。集合覆盖问题要找到S的一个最小的子集，使得他们的并集等于全集。

SAT Algorithm 1b: Deterministic

Let $I = \{\}$. Repeat:

1. If I falsifies some $c \in \phi$: return NO
2. Select a variable x such that $x, \bar{x} \notin I$
3. If there is none: return YES
4. Let $I = I \cup \{x\}$ or $I = I \cup \{\neg x\}$

Let $I = \{\}$. Repeat:

1. If I falsifies some $c \in \phi$:
   - 1.1 Backtrack to the last decision $\neg x$ — undo last decisions
   - 1.2 If there is none: return NO
   - 1.3 Let $I = I \cup \{x\}$ — flip last negative decision
2. Else:
   - 2.1 Select a variable x such that $x, \bar{x} \notin I$
   - 2.2 If there is none: return YES
   - 2.3 Let $I = I \cup \{\neg x\}$ — make a decision

利用backtrack规避无意义的全局搜索，增加搜索效率

Unit Propagation

Definition: closure of I under unit propagation relative to ϕ

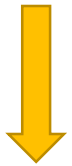
- Let $I^0 = I$
- Repeat for $j > 0$ until $I^j = I^{j+1}$:
  - If there is a $(x_1 \vee \ldots \vee x_k) \in \phi$ with $\bar{x}_1, \ldots, \bar{x}_k \in I^j$:
    Return conflict $(x_1 \vee \ldots \vee x_k)$
  - If there is a $(x_1 \vee \ldots \vee x_{k+1}) \in \phi$ with $\bar{x}_1, \ldots, \bar{x}_k \in I^j$:
    Let $I^{j+1} = I^j \cup \{x_{k+1}\}$
- Return I^j

Ex. 1: $I = \{\neg p\}$ $\phi = \{(\neg p \vee \neg q \vee s), (p \vee \neg q \vee r), (p \vee q)\}$

Unit Propagation

Ex. 1: $I = \{\neg p\}$ $\phi = \{(\neg p \vee \neg q \vee s), (p \vee \neg q \vee r), (p \vee q)\}$

- $I^0 = \{\neg p\}$



Ex. 1: $I = \{\neg p\}$ $\phi = \{(\neg p \vee \neg q \vee s), (p \vee \neg q \vee r), (p \vee q)\}$

- $I^0 = \{\neg p\}$
- $I^1 = \{\neg p, q\}$

- $I^2 = \{\neg p, q, r\}$
- Conflict with $(p \vee \neg q)$

Watched-Literal Scheme

For every $c \in \phi$, select two distinct watched literals $x_c, y_c \in c$
such that $\bar{x}_c \notin I$ if possible, otherwise choose arbitrarily
and $\bar{y}_c \notin I$ if possible, otherwise choose arbitrarily.

从每个clause里面选取两个候选的literal，即watched literals

How to close I under UP relative to ϕ?

For every $(x_1 \vee \ldots \vee x_k) \in \phi$ with watched literals x_c, y_c:

(i) If $\bar{x}_c \in I, \bar{y}_c \in I$: Try to update x_c, y_c. Otherwise return conflict.

(ii) If $\bar{x}_c \in I, \bar{y}_c \notin I$: Try to update x_c. Otherwise add y_c to I.

If $\bar{x}_c \notin I, \bar{y}_c \in I$: Try to update y_c. Otherwise add x_c to I.

=

For every $(x_1 \vee \ldots \vee x_k) \in \phi$ that watches $\bar{z}_1 = x_c$ (w.l.o.g.):

1. Try to update x_c.
2. Otherwise: If $\bar{y}_c \in I'$: Return conflict.
   If $\bar{y}_c \notin I'$: Add y_c to I'.

Now $I \cup \{z_1\}$ is closed under UP relative to ϕ. Repeat for z_2, and so on.

Watched-Literal Scheme

Clauses: $p \vee q \vee s$ $\quad \neg r \vee \neg s \vee \neg t$

$t \vee \neg u$ $\quad t \vee \neg v$ $\quad t \vee \neg w$

$u \vee v \vee w \vee y$ $\quad v \vee \neg y$

Decisions: $\neg p, \neg q, r$

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$

Watched-Literal Scheme

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$							
$\bar{q}$	s, q						
s							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					

Watched-Literal Scheme

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r		$\bar{r}, \bar{t}$					
$\bar{t}$							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r		$\bar{r}, \bar{t}$					
$\bar{t}$			$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$		
$\bar{u}$							
$\bar{v}$							
$\bar{w}$							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r		$\bar{r}, \bar{t}$					
$\bar{t}$			$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$		
$\bar{u}$						y, v	
$\bar{v}$							
$\bar{w}$							

Watched-Literal Scheme

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r		$\bar{r}, \bar{t}$					
$\bar{t}$			$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$		
$\bar{u}$						y, v	
$\bar{v}$						y, v	
$\bar{w}$							
y							

I	Clauses and Watched Literals						
	$p \vee q \vee s$	$\bar{r} \vee \bar{s} \vee \bar{t}$	$t \vee \bar{u}$	$t \vee \bar{v}$	$t \vee \bar{w}$	$u \vee v \vee w \vee y$	$v \vee \bar{y}$
	p, q	$\bar{r}, \bar{s}$	$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$	u, v	$v, \bar{y}$
$\bar{p}$	s, q						
$\bar{q}$	s, q						
s		$\bar{r}, \bar{t}$					
r		$\bar{r}, \bar{t}$					
$\bar{t}$			$t, \bar{u}$	$t, \bar{v}$	$t, \bar{w}$		
$\bar{u}$						y, v	
$\bar{v}$						y, v	$v, \bar{y}$ ↯
$\bar{w}$							
y							

Conflict-Driven Clause Learning

- Algorithm 2 with Watched-Literal Scheme still spends almost all its time on unit propagation.
- Suppose $I = \{x_1, \ldots, x_k\}$ leads to a conflict.
  That is: I falsifies some $c \in \phi$.
- Let's learn from the conflict to avoid similar mistakes later:
  - Find a cause $\{x_{i_1}, \ldots, x_{i_l}\} \subseteq I$ of the conflict.
  - Add learnt clause $(\bar{x}_{i_1} \vee \ldots \vee \bar{x}_{i_l})$ to avoid the conflict next time!
  - This avoids assignments $x_{i_1}, \ldots, x_{i_l}$ in the remaining search.
  - Note: must have $\phi \models (\bar{x}_{i_1} \vee \ldots \vee \bar{x}_{i_l})$.

Let $I = \{\}$. Repeat:

1. Close I under unit propagation relative to ϕ
2. If conflict:
   - 2.1 Analyse conflict — find the cause
   - 2.2 Backtrack to the appropriate level — undo last decisions
   - 2.3 If there is none: return NO
   - 2.4 Add conflict clause to ϕ — new decision implicitly
3. Else:
   - 3.1 Select a variable x such that $x, \bar{x} \notin I$
   - 3.2 If there is none: return YES
   - 3.3 Let $I = I \cup \{\neg x\}$ — make a decision

通过把冲突的clause加入到ϕ当中来避免下次重复尝试

例如如果a和b同时为真时产生冲突，则把$\bar{a} \vee \bar{b}$加入到ϕ中，
这样就能强迫程序下次求解时避开a和b同时为真的解

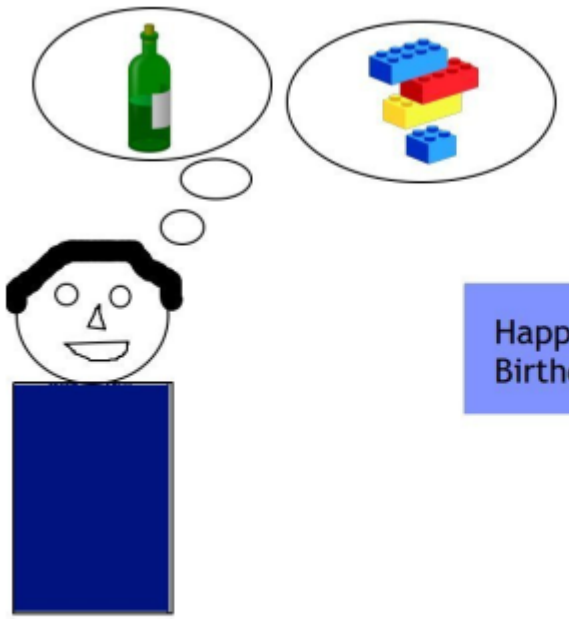
Knowledge and Action

Motivation

Observation: Non-knowledge is important

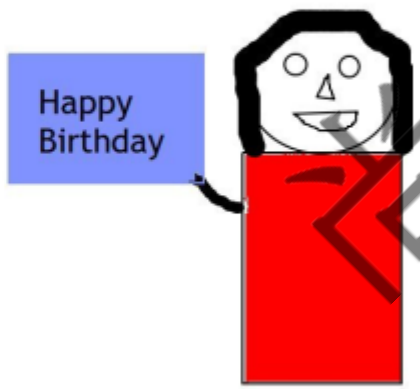
Not only what we *know* is relevant, but also what we *don't know*.

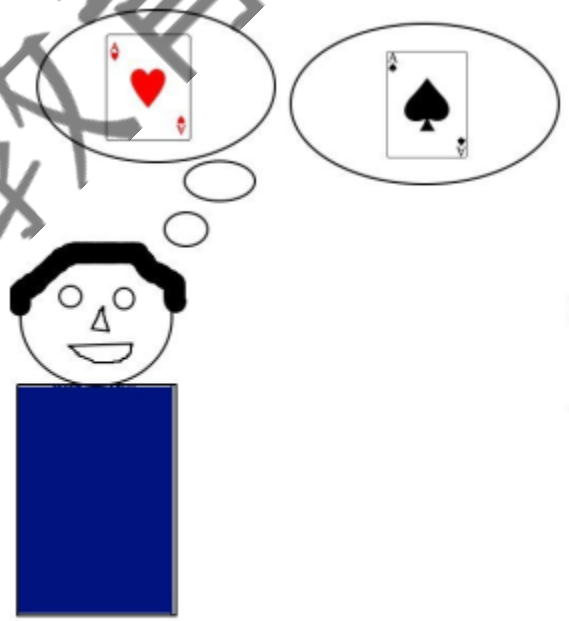
如何描述“不知道”这个概念呢



You don't know what's in the gift box.

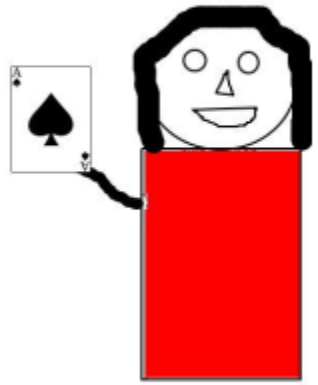
So you'll treat it with great care.





You know Jane is holding ace of spades *or* of hearts, but not which.

So you'll look for a strategy that wins in either case.



Knowledge and Action

Syntax:

- Atomic propositions
- Negation: $\neg\phi$
- Disjunction: $(\phi_1 \vee \phi_2)$
- Conjunction: $(\phi_1 \wedge \phi_2)$
- Knowledge: $\mathbf{K}\phi$

Examples:

1. $\mathbf{K}(p \vee \neg p)$
2. $\mathbf{K}(p \vee q) \rightarrow (\mathbf{K}p \vee \mathbf{K}q)$
3. $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
4. $p \rightarrow \mathbf{K}p$
5. $(\mathbf{K}p \vee \mathbf{K}q) \rightarrow \mathbf{K}(p \vee q)$

Possible-Worlds Semantics

- In propositional and first-order logic, an interpretation is a one possibility what the real world could be.
- Incomplete knowledge:
  - The agent does not know everything about the real world.
  - So the agent considers multiple worlds possible.
- Possible worlds semantics of knowledge:
  - The agent considers a set of worlds possible
  - Knowledge is what is true in all possible worlds.

只有我们确定知道的东西才是Knowledge

因此我们要考查所有可能出现的情况

Knowledge and Action

- Example: $\mathbf{K}(p \vee q) \wedge \neg\mathbf{K}p \wedge \neg\mathbf{K}q$
  - $\mathbf{K}(p \vee q)$: it is known that $(p \vee q)$.
  - $\neg\mathbf{K}p$: it is not known that p.
  - $\neg\mathbf{K}q$: it is not known that q.

  ⇕

  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: not all possible worlds satisfy p.
  - $\neg\mathbf{K}q$: it is not known that q.

  ⇕

  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world falsifies p.
  - $\neg\mathbf{K}q$: it is not known that q.

  ⇕

  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world satisfies $\neg p$.
  - $\neg\mathbf{K}q$: it is not known that q.

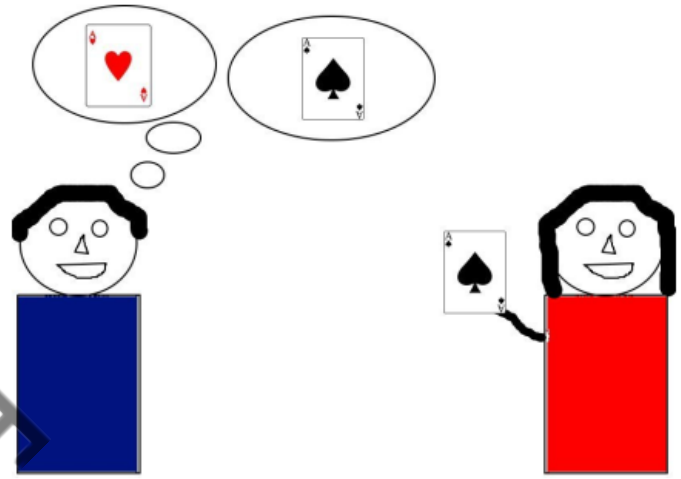
  - $\mathbf{K}(p \vee q)$: every possible world satisfies p or q.
  - $\neg\mathbf{K}p$: some possible world satisfies $\neg p$.
  - $\neg\mathbf{K}q$: some possible world satisfies $\neg q$.

  So there are at least two possible worlds: $\{p, \neg q\}$ and $\{\neg p, q\}$

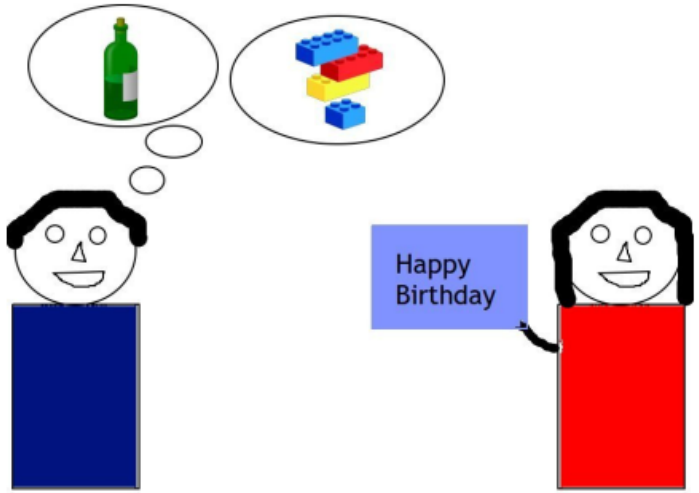
Knowledge and Action

Semantics:

- A world is a complete and consistent set of propositions.
- An interpretation is a pair e, w of a set of possible worlds e and a real world w.
- e, w satisfies p iff $p \in w$
- e, w satisfies $\neg\phi$ iff e, w does not satisfy ϕ
- e, w satisfies $(\phi_1 \vee \phi_2)$ iff e, w satisfies ϕ_1 or ϕ_2
- e, w satisfies $(\phi_1 \wedge \phi_2)$ iff e, w satisfies ϕ_1 and ϕ_2
- e, w satisfies $\mathbf{K}\phi$ iff for all $w' \in e$: e, w' satisfies ϕ



$$\mathbf{K}((\spadesuit \vee \heartsuit) \wedge \neg\mathbf{K}\spadesuit \wedge \neg\mathbf{K}\heartsuit)$$



$$\mathbf{K}\exists x\,(\mathrm{InBox}(x) \wedge \neg\mathbf{K}\mathrm{InBox}(x))$$

Logical Omniscience and Complexity

- An agent with knows what is logically entailed by its knowledge base.
  - Let KB be the agent's knowledge base.
  - Then ϕ is known iff KB entails ϕ.

  The agent is logically omniscient.

从knowledge base中可以得到的所有事实agent都应该知道

但推出这些所有事实并不容易

- Complexity of "is ϕ known?":
  - Classical propositional logic: co-NP-complete
  - Classical first-order logic: undecidable

- Classical logic is an inappropriate model of human knowing:
  - Complexity of classical logic is beyond human capabilities.
  - Most of the time, humans only use fragments of classical logic.

- Satisfiability is NP-complete in propositional logic.
- Validity is co-NP-complete in propositional logic.
- Satisfiability/validity is undecidable in first-order logic.

回忆一下
Satisfiability – 有成立的可能
Validity – 任何情况下都成立

Recall: ϕ is satisfiable iff $\neg\phi$ is not valid
ϕ is valid iff $\neg\phi$ is not satisfiable

Actions and Change

An action can only be executed under certain circumstances.

The Qualification Problem

Represent the preconditions (qualifications) of an action.

Ex.: You want to take a bus b to get to a destination d.
What must be true for this to be possible?

动作的发生需要有前置条件

Most things do not change when an action is executed.

The Frame Problem

Represent what is left unchanged by an action (frame axioms).

Ex.: You don't magically disappear from the bus when it moves.
The weather also remains unchanged when the bus moves.

现实中任何状态都有持续性，在逻辑中我们也希望能表达这种“惯性”

State constraints must be satisfied over the course of actions.

The Ramification Problem

Represent indirect effects caused by state constraints.

Ex.: If you're on the bus, your location is where the bus is.
You cannot be at two busses at once.

状态需要被限制，比如你不能同时在两个不同的地方

Actions and Change

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Single Actions: $[a]\phi$
- Action Sequence: $\Box\,\phi$

- You don't fall off the bus when the bus moves:
  $\Box \forall b_1 \forall b_2 \forall d \left(\mathrm{On}(b_1) \rightarrow [\mathrm{goTo}(b_2, d)]\mathrm{On}(b_1)\right)$

- You cannot be on two busses at once:
  $\Box \forall b_1 \forall b_2 \left(b_1 \neq b_2 \rightarrow \neg\mathrm{On}(b_1) \vee \neg\mathrm{On}(b_2)\right)$

Syntax:

- Terms: variables x, functions $f(t_1, \ldots, t_k)$
- Equality: $t_1 = t_2$
- Predicates: $P(t_1, \ldots, t_k)$
- Single Actions: $[a]\phi$
- Action Sequence: $\Box\,\phi$
- Existential quantification: $\exists x\,\phi$
- Universal quantification: $\forall x\,\phi$
- Negation: $\neg\phi$
- Disjunction: $(\phi_1 \vee \phi_2)$
- Conjunction: $(\phi_1 \wedge \phi_2)$

Actions and Change

Solving the Frame Problem with Successor-State Axioms

When are we on a bus?

What are the effects?

$\Box \forall b\, [\text{getOn}(b)]\text{On}(b)$

$\Box \forall b\, [\text{getOff}]\neg\text{On}(b)$

$\Box \forall a \forall b\, (a = \text{getOn}(b) \rightarrow [a]\text{On}(b))$

$\Box \forall a \forall b\, (a = \text{getOff} \rightarrow [a]\neg\text{On}(b))$

Assume causal completeness:

$\Box \forall a \forall b\, (\neg\text{On}(b) \wedge [a] \quad \text{On}(b) \rightarrow a = \text{getOn}(b))$

$\Box \forall a \forall b\, (\quad \text{On}(b) \wedge [a]\neg\text{On}(b) \rightarrow a = \text{getOff})$

These axioms are equivalent to a single successor-state axiom:

$\Box \forall a \forall b\, ([a]\text{On}(b) \leftrightarrow a = \text{getOn}(b) \vee (\text{On}(b) \wedge a \neq \text{getOff}))$

That is: What's true *after* a is fully determined by what's true *before* a.

Successor-state axiom表示了当前状态为真时，其前置的动作应该是什么，就是知道了结果时倒推其可能的原因

Actions and Change

Solving the Frame Problem with Successor-State Axioms

Definition: successor-state axiom

A **successor-state axiom** has the form

$$\Box \forall a \forall x_1 \ldots \forall x_k \left([a]P(x_1, \ldots, x_k) \leftrightarrow \gamma_P \right)$$

where γ_P does not mention $\Box$ or $[A]$ operators.

Typical form of γ_P is $\gamma_P^+ \vee (P(x_1, \ldots, x_k) \wedge \neg \gamma_P^-)$:

- γ_P^+ is the positive effect condition
- γ_P^- is the negative effect condition

- You're on a bus iff you got on it *or* you were on it and didn't get off it:

$$\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee (\mathrm{On}(b) \wedge a \neq \mathrm{getOff}) \right)$$

- You're at position is p iff you were on a bus that moved to p *or* you were at p already and not on a bus that moved:

$$\Box \forall a \forall p \left([a]\mathrm{At}(p) \leftrightarrow \exists b \left(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b) \right) \vee \left(\mathrm{At}(p) \wedge \neg \exists d \exists b \left(a = \mathrm{goTo}(b, d) \wedge \mathrm{On}(b) \right) \right) \right)$$

Projection

Definition: projection

Projection is the following decision problem:
Input: formulas ϕ, ψ, successor-state axioms Σ, actions $A_1, \ldots, A_\ell$.
Problem: do ϕ and Σ entail ψ after $A_1, \ldots, A_\ell$?

$$\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi?$$

Ex.: $\mathrm{At}(\mathrm{Central}) \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})?$

There are two approaches to projection problems:

- Regression: reduce to $\phi \models \psi^*$
- Progression: reduce to $\phi^* \models \psi$

Both reduce projection to ordinary entailment without actions.

Regression

- Successor state axioms relate truth after a to truth before a:
  $$\Box \forall a \forall x_1 \ldots \forall x_k \left([a]P(x_1, \ldots, x_k) \leftrightarrow \gamma_P \right)$$
- Let's define a procedure $\mathcal{R}(\phi)$ which iteratively replaces $[A]P(t_1, \ldots, t_k)$ with $\gamma_P \begin{smallmatrix} a\ x_1 \ldots x_k \\ A\ t_1 \ldots t_k \end{smallmatrix}$.
- $\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi$ iff $\phi \models \mathcal{R}([A_1] \ldots [A_\ell]\psi)$.

目的是去掉所有的动作，把问题还原为最简单的逻辑形式从而求解

Theorem: regression

Let ϕ, ψ be without $[A]$ and $\Box$.
Let $A_1, \ldots, A_\ell$ be ground terms.
Let Σ be a set of successor-state axioms.
Then:
$$\phi \wedge \Sigma \models [A_1] \ldots [A_\ell]\psi \text{ iff } \phi \models \mathcal{R}([A_1] \ldots [A_\ell]\psi)$$

- $\mathcal{R}([A_1] \ldots [A_k](\alpha \vee \beta)) = (\mathcal{R}([A_1] \ldots [A_k]\alpha) \vee \mathcal{R}([A_1] \ldots [A_k]\beta))$
- $\mathcal{R}([A_1] \ldots [A_k](\alpha \wedge \beta)) = (\mathcal{R}([A_1] \ldots [A_k]\alpha) \wedge \mathcal{R}([A_1] \ldots [A_k]\beta))$
- $\mathcal{R}([A_1] \ldots [A_k]\neg\alpha) = \neg\mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]\exists x\, \alpha) = \exists x \mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]\forall x\, \alpha) = \forall x \mathcal{R}([A_1] \ldots [A_k]\alpha)$
- $\mathcal{R}([A_1] \ldots [A_k]t_1 = t_2) = t_1 = t_2$
- $\mathcal{R}(P(t_1, \ldots, t_k)) = P(t_1, \ldots, t_k)$
- $\mathcal{R}([A_1] \ldots [A_{\ell+1}]P(t_1, \ldots, t_k)) = \mathcal{R}([A_1] \ldots [A_\ell]\gamma_P \begin{smallmatrix} a & x_1 \ldots x_k \\ A_{\ell+1} & t_1 \ldots t_k \end{smallmatrix})$
  where
  - γ_P is the right-hand side of the successor-state axiom of P and
  - variables in γ_P are renamed to avoid clashes with variables in $A_{\ell+1}$

Regression

Let Σ contain the successor-state axioms for $\mathrm{On}(b)$, $\mathrm{At}(p)$.

$$\phi \wedge \Sigma \models [\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni})\ ?$$

$$\Box \forall a \forall p \left([a]\mathrm{At}(p) \leftrightarrow \exists b \left(a = \mathrm{goTo}(b, p) \wedge \mathrm{On}(b)\right) \vee \left(\mathrm{At}(p) \wedge \neg \exists d \exists b \left(a = \mathrm{goTo}(b, d) \wedge \mathrm{On}(b)\right)\right)\right)$$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})][\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni})]\mathrm{At}(\mathrm{Uni}))$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\gamma_{\mathrm{At}}{}^{a}_{\mathrm{goTo}(\mathrm{M50},\mathrm{Uni})}{}^{p}_{\mathrm{Uni}})$

iff $\phi \models \mathcal{R}([\mathrm{getOn}(\mathrm{M50})](\exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathrm{On}(b)\right)) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}([\mathrm{getOn}(\mathrm{M50})]\mathrm{On}(b))\right) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \mathcal{R}(\gamma_{\mathrm{On}}{}^{a}_{\mathrm{getOn}(\mathrm{M50})}{}^{b}_{b})\right) \vee \ldots$

iff $\phi \models \exists b \left(\mathrm{goTo}(\mathrm{M50}, \mathrm{Uni}) = \mathrm{goTo}(b, \mathrm{Uni}) \wedge \left(\mathrm{getOn}(\mathrm{M50}) = \mathrm{getOn}(b) \vee \left(\mathcal{R}(\mathrm{On}(b)) \wedge \mathrm{getOn}(\mathrm{M50}) \neq \mathrm{getOff}\right)\right)\right) \vee \ldots$

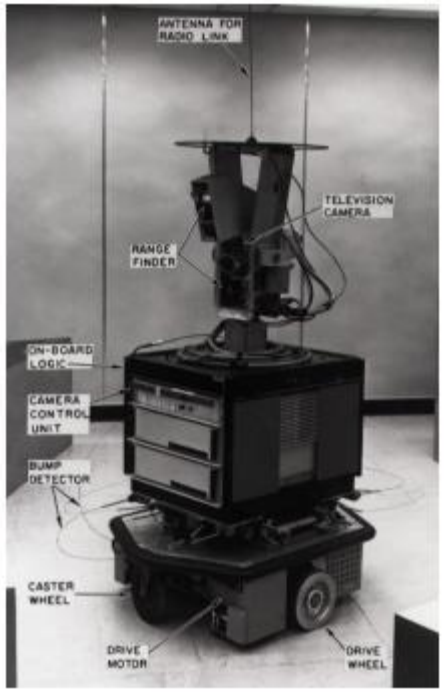
$$\Box \forall a \forall b \left([a]\mathrm{On}(b) \leftrightarrow a = \mathrm{getOn}(b) \vee \left(\mathrm{On}(b) \wedge a \neq \mathrm{getOff}\right)\right)$$

iff $\phi \models \exists b \underbrace{\left(\mathrm{M50} = b \wedge \left(\mathrm{M50} = b \vee \mathcal{R}(\mathrm{On}(b))\right)\right)} \vee \ldots$ ✓

Valid if b is M50. So the whole formula is valid and hence entailed by ϕ.

STRIPS Planning

Stanford Research Institute Problem Solver



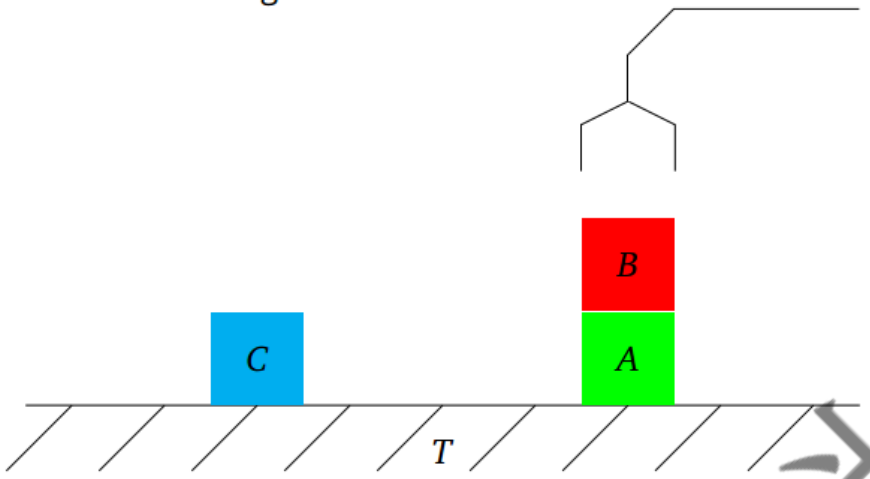
- A state is a set of propositions.
  - Propositions in the current state are true.
  - Propositions not in the current state are false.
- Action operators consist of:
  - Precondition: propositions that must be true
  - Positive effects: propositions that are added to state
  - Negative effects: propositions that are deleted from state

A problem instance consists of:

- Initial state
- Goal state
- Set of action operators

STRIPS Planning Stanford Research Institute Problem Solver

A robot is stacking blocks on a table:



- Initial state: Block(A), Block(B), Block(C), HandEmpty, On(C, T), Clear(C), On(A, T), On(B, A), Clear(B)
- Action pickUp(x, y) picks up block x when it is on y:
  - Precondition: Block(x), On(x, y), Clear(x), HandEmpty
  - Positive effects: Holding(x), ¬Clear(x), Clear(y)
  - Negative effects: On(x, y), HandEmpty
- Action putOn(x, y) puts block x on (block or table) y:
  - Precondition: Block(x), Holding(x), Clear(y)
  - Positive effects: On(x, y), Clear(x), HandEmpty
  - Negative effects: Holding(x), Clear(y)
- Action putOnTable(x) puts x on the table:
  - Precondition: Block(x), Holding(x)
  - Positive effects: On(x, T), Clear(x), HandEmpty
  - Negative effects: Holding(x)

Example

Consider the following basic action theory, where γ and φ are the right-hand sides of the successor-state axiom of Sick and the axiom for SF from the previous task.

$$\begin{aligned}
\Sigma_0 &= \{\text{Sick} \wedge \neg\text{Placebo}(\#1) \wedge \text{Placebo}(\#2)\} \\
\Sigma_1 &= \{\text{TRUE}\} \\
\Sigma_{\text{dyn}} &= \{\Box[a]\text{Sick} \leftrightarrow \gamma, \\
&\quad \Box[a]\text{Placebo}(x) \leftrightarrow \text{Placebo}(x), \\
&\quad \Box\text{Poss}(a) \leftrightarrow \text{TRUE}, \\
&\quad \Box\text{SF}(a) \leftrightarrow \varphi\}
\end{aligned}$$

Example

Prove that $\Sigma_0 \wedge \Sigma_{\text{dyn}} \models [\text{take}(\#1)]\neg\text{Sick}$ using regression.

$$\Sigma_0 \wedge \Sigma_{\text{dyn}} \models [\text{take}(\#1)]\neg\text{Sick} \;\text{ iff }\; \Sigma_0 \models \mathcal{R}[\langle\rangle, [\text{take}(\#1)]\neg\text{Sick}]$$

$$\begin{aligned}
& \mathcal{R}[\langle\rangle, [\text{take}(\#1)]\neg\text{Sick}] \\
= \;& \mathcal{R}[\text{take}(\#1), \neg\text{Sick}] \\
= \;& \neg\mathcal{R}[\text{take}(\#1), \text{Sick}] \\
= \;& \neg\mathcal{R}[\langle\rangle, \gamma^{a}_{\text{take}(\#1)}] \\
= \;& \neg\mathcal{R}[\langle\rangle, (\text{Sick} \wedge \forall x\,(a \neq \text{take}(x) \vee \text{Placebo}(x)))^{a}_{\text{take}(\#1)}]
\end{aligned}$$

Example

Prove that $\Sigma_0 \wedge \Sigma_{\text{dyn}} \models [\text{take}(\#1)]\neg\text{Sick}$ using regression.

$$\Sigma_0 \wedge \Sigma_{\text{dyn}} \models [\text{take}(\#1)]\neg\text{Sick} \;\text{ iff }\; \Sigma_0 \models \mathcal{R}[\langle\rangle, [\text{take}(\#1)]\neg\text{Sick}]$$

$$\begin{aligned}
&= \neg\mathcal{R}[\langle\rangle, (\text{Sick} \wedge \forall x\,(\text{take}(\#1) \neq \text{take}(x) \vee \text{Placebo}(x)))] \\
&= \neg(\mathcal{R}[\langle\rangle, \text{Sick}] \wedge \forall x\,(\mathcal{R}[\langle\rangle, \text{take}(\#1) \neq \text{take}(x)] \vee \mathcal{R}[\langle\rangle, \text{Placebo}(x)])) \\
&= \neg(\text{Sick} \wedge \forall x\,(\text{take}(\#1) \neq \text{take}(x) \vee \text{Placebo}(x))) \\
&\overset{\text{simpl}}{=} (\neg\text{Sick} \vee \neg\forall x\,(\text{take}(\#1) \neq \text{take}(x) \vee \text{Placebo}(x))) \\
&\overset{\text{simpl}}{=} (\neg\text{Sick} \vee \exists x\,(\text{take}(\#1) = \text{take}(x) \wedge \neg\text{Placebo}(x))) \\
&\overset{\text{simpl}}{=} (\neg\text{Sick} \vee \exists x\,(\#1 = x \wedge \neg\text{Placebo}(x))) \\
&\overset{\text{simpl}}{=} (\neg\text{Sick} \vee \neg\text{Placebo}(\#1))
\end{aligned}$$

Exercise

Suppose we have a domestic service robot cleaning up our house. It has a box, and into that box it can put objects. The robot can also shake the box to test whether something is in it. Finally it can move the box and its contents to other locations. We assume that in reality, the box does contain some object, but the robot does not know that. This scenario is modelled by the following basic action theory:

$$\Sigma_0 = \{\exists x\, \mathrm{InBox}(x)\}$$

$$\Sigma_{\mathrm{dyn}} = \{\Box \mathrm{Poss}(a) \leftrightarrow \mathrm{TRUE}$$

$$\Box[a]\mathrm{InBox}(x) \leftrightarrow a = \mathrm{putInBox}(x) \vee \mathrm{InBox}(x)$$

$$\Box[a]\mathrm{location}(x) = y \leftrightarrow (a = \mathrm{moveBox}(y) \wedge \mathrm{InBox}(x)) \vee (\forall y'\, a \neq \mathrm{moveBox}(y') \wedge \mathrm{location}(x) = y)\}$$

Exercise

Prove or disprove the following projection problems using regression:

(a) $\Sigma_0 \wedge \Sigma_{\text{dyn}} \models \forall x\,[\text{putInBox}(x)]\text{InBox}(x)$

(b) $\Sigma_0 \wedge \Sigma_{\text{dyn}} \models \forall y\,[\text{moveBox}(y)]\forall x\,(\text{InBox}(x) \rightarrow \text{location}(x) = y)$

Questions and good luck