Logical Agents

吉建民

USTC
jianmin@ustc.edu.cn

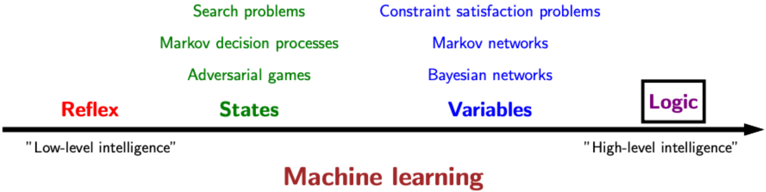
2024 年 3 月 31 日

Used Materials

Disclaimer: 本课件采用了 S. Russell and P. Norvig's Artificial Intelligence –A modern approach slides, 徐林莉老师课件和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

Some modeling paradigms

- State-based models: search problems, MDPs, games
  - Applications: routing finding, game playing, etc.
  - Think in terms of **states**, **actions**, and **costs**
- Variable-based models: CSPs, Bayesian networks
  - Applications: scheduling, medical diagnosis, etc.
  - Think in terms of **variables** and **factors**
- Logic-based models: propositional logic, first-order logic
  - Applications: theorem proving, verification, reasoning
  - Think in terms of **logical formulas** and **inference rules**



Example

- Question: If $X_1 + X_2 = 10$ and $X_1 - X_2 = 4$, what is X_1?
- Think about how you solved this problem. You could treat it as a CSP with variables X_1 and X_2, and search through the set of candidate solutions, checking the constraints.
- However, more likely, you just added the two equations, divided both sides by 2 to easily find out that $X_1 = 7$.
- This is the power of logical inference, where we apply a set of truth-preserving rules to arrive at the answer. This is in contrast to what is called model checking, which tries to directly find assignments.
- We'll see that logical inference allows you to perform very powerful manipulations in a very compact way. This allows us to vastly increase the representational power of our models.

A historical note

- Logic was dominant paradigm in AI before 1990s
- Problem 1: deterministic, didn't handle uncertainty (probability addresses this)
- Problem 2: rule-based, didn't allow fine tuning from data (machine learning addresses this)
- Strength: provides expressiveness in a compact way
  - There is one strength of logic which has not quite yet been recouped by existing probability and machine learning methods, and that is expressivity of the model

Motivation: smart personal assistant

Tell information →  ← **Ask** questions

Use natural language!

- How to build smart personal assistants?
  - Systems like Apple's Siri, Microsoft Cortana, Amazon Echo (Alexa), and Google Now (Assistant)
  - Smart speaker (~~current~~ past): Intent Detection + Slot Filling + Search
  - Smart speaker (~~future~~ current): Large Language Model (LLM) + Reasoning
- Need to:
  - Digest **heterogenous** information
  - Reason **deeply** with that information

Language

- **Language** is a mechanism for expression
- Natural languages (informal):
  - 汉语：二能除偶数。
  - English: Two divides even numbers.
- Programming languages (formal):
  - Python: `def even(x): return x % 2 == 0`
  - C++: `bool even(int x) { return x % 2 == 0; }`
- Logical languages (formal):
  - First-order logic: $\forall x.\ \mathsf{Even}(x) \rightarrow \mathsf{Divides}(x, 2)$

Two goals of logic

- Represent knowledge about the world

- Reason with that knowledge

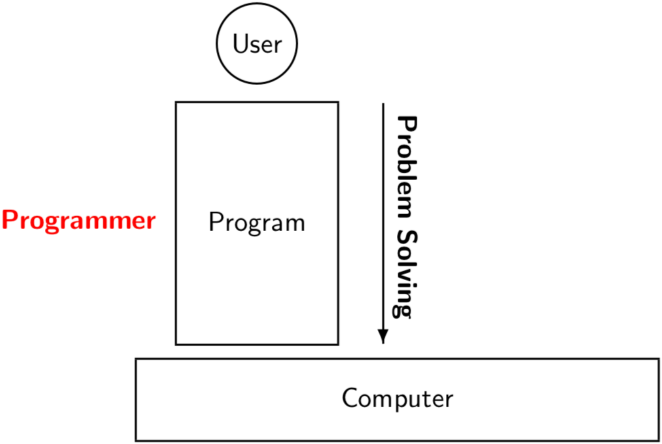
Elaboration Tolerance

- Elaboration Tolerance (McCarthy, 1998)

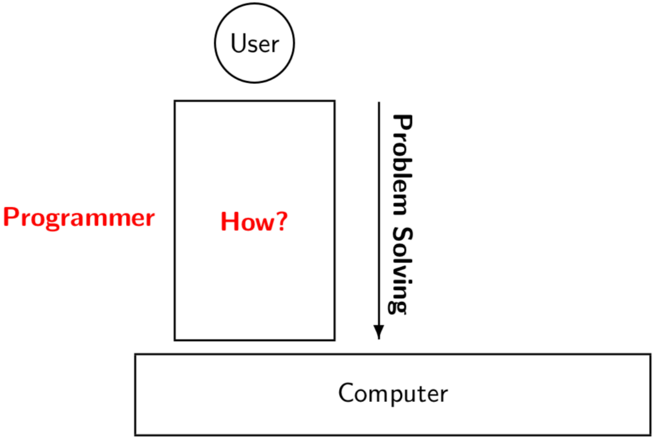
  "A formalism is elaboration tolerant [if] it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances."

- Uniform problem representation
  For solving a problem instance I of a problem class C,
  - I is represented as a set of facts P_I,
  - C is represented as a set of rules P_C, and
  - P_C can be used to solve all problem instances in C

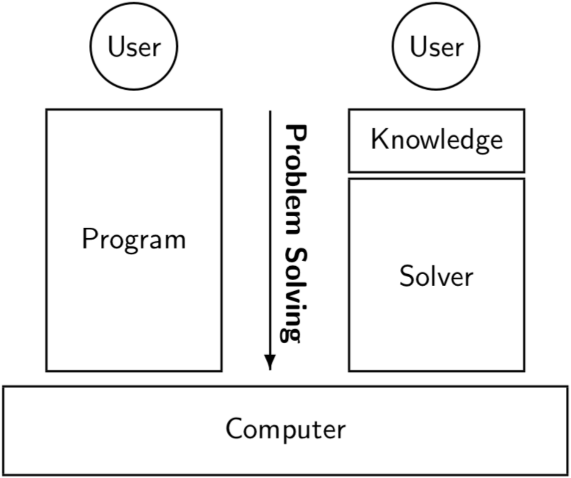
Traditional Software



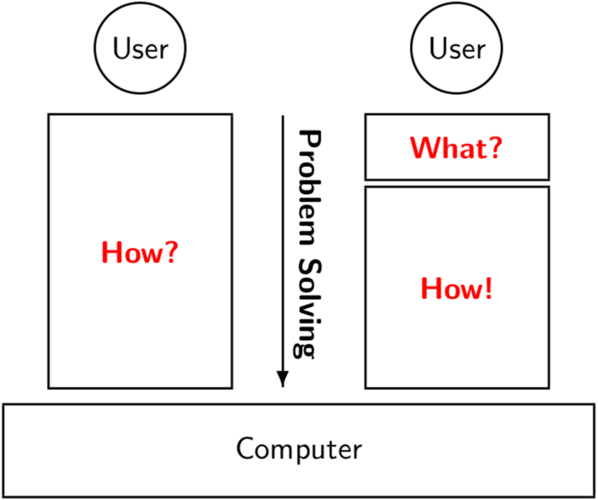
Traditional Software



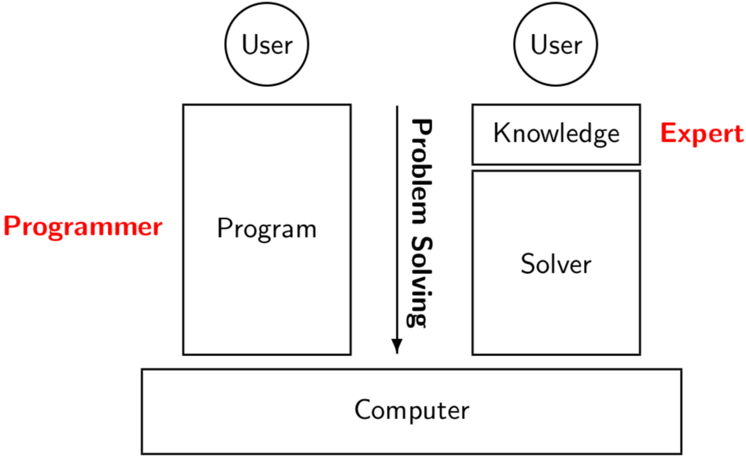
Knowledge-driven Software



Knowledge-driven Software

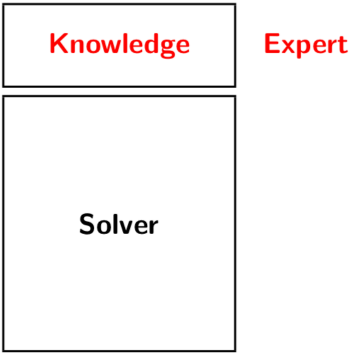


Knowledge-driven Software



What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability



What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

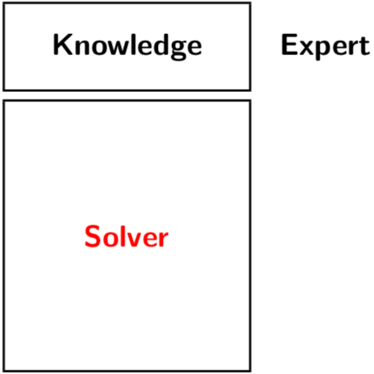
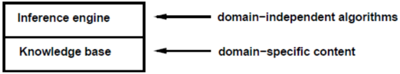


Table of Contents

Knowledge bases



Knowledge base（知识库） = set of sentences in a formal language

将新语句添加到知识库——
Declarative approach to building an agent (or other system):
  TELL（告诉） it what it needs to know

查询目前所知内容——
Then it can ASK（询问） itself what to do — answers should follow from the KB

Agents can be viewed at the knowledge level（知识层）
  i.e., what they know, regardless of how implemented

Or at the implementation level（实现层）
  i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

- TELL⟶ASK⟶TELL
- 表示语言的细节隐含于MAKE-PERCEPT-SENTENCE和MAKE-ACTION-QUERY中
- 推理机制的细节隐藏于TELL和ASK中

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
           t, a counter, initially 0, indicating time

   TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
   action ← ASK(KB, MAKE-ACTION-QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

The agent must be able to:

Represent states, actions, etc.	表示状态和行为
Incorporate new percepts	加入新的感知信息
Update internal representations of the world	更新关于世界的状态表示
Deduce hidden properties of the world	推导关于世界的隐藏信息
Deduce appropriate actions	推导应采取的合适的行为

Table of Contents

Wumpus World PEAS description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

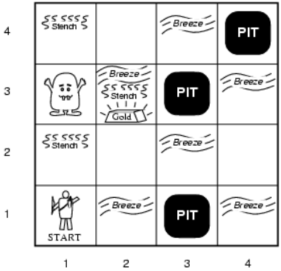
Environment

4×4网格

智能体初始在[1,1]，面向右方

金子和wumpus在[1,1]之外随机均匀分布

[1,1]之外的任意方格是陷阱的概率是0.2



Actuators Left turn, Right turn,

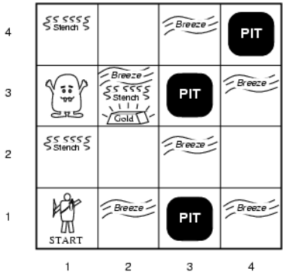
Forward, Grab, Shoot

- 智能体可向前、左转或右转
- 智能体如果进入一个有陷阱或者活着的wumpus的方格，将死去。
- 如果智能体前方有一堵墙，那么向前移动无效
- Grab：捡起智能体所在方格中的一个物体
- Shoot：向智能体所正对方向射箭（只有一枝箭）

Wumpus World PEAS description

Sensors

- Smell: 在wumpus所在之处以及与之直接相邻的方格内，智能体可以感知到臭气。
- Breeze: 在与陷阱直接相邻的方格内，智能体可以感知到微风。
- Glitter(发光): 在金子所处的方格内，智能体可以感知到闪闪金光。
- 当智能体撞到墙时，它感受到撞击。
- 当wumpus被杀死时，它发出洞穴内任何地方都可感知到的悲惨嚎叫。



以5个符号的列表形式将感知信息提供给智能体，例如(stench, breeze, none, none, none)。

Wumpus world characterization

Complete?? No —only local perception

Deterministic?? Yes —outcomes exactly specified

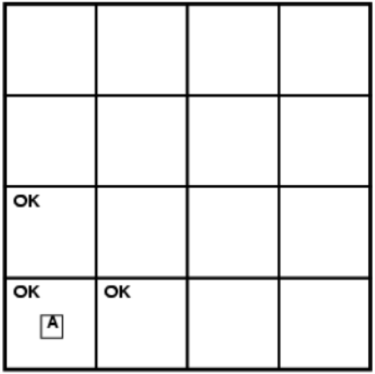
Episodic?? No —sequential at the level of actions

Static?? Yes —Wumpus and Pits do not move

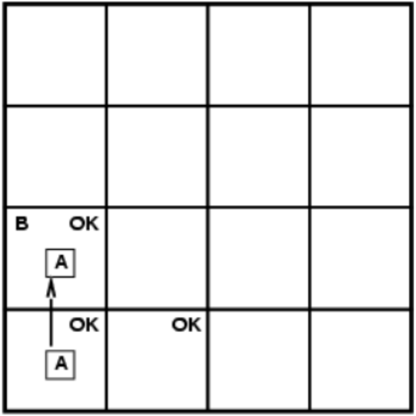
Discrete?? Yes

Single-agent?? Yes —Wumpus is essentially a natural feature

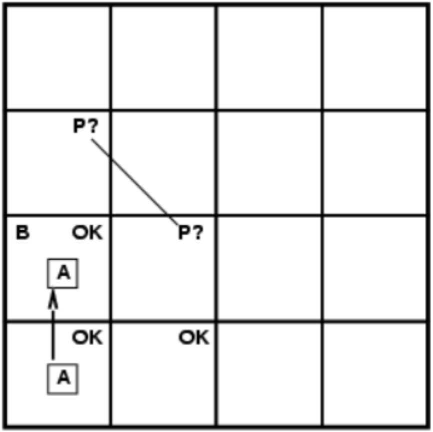
Exploring a wumpus world



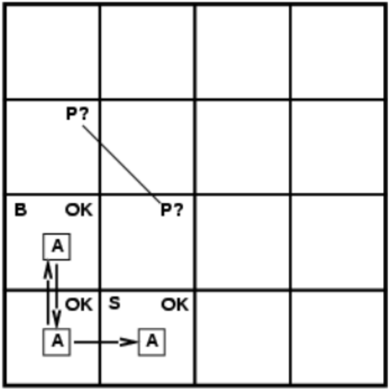
Exploring a wumpus world



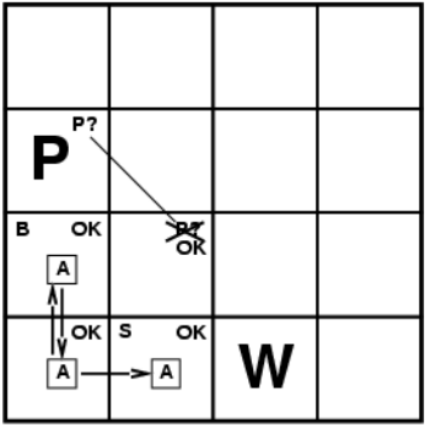
Exploring a wumpus world



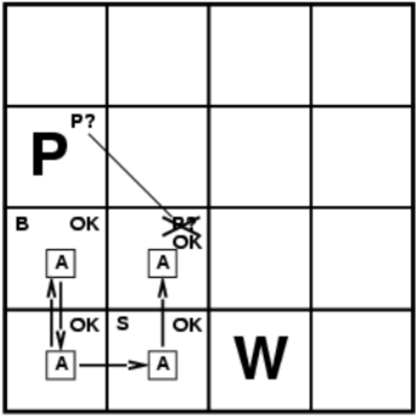
Exploring a wumpus world



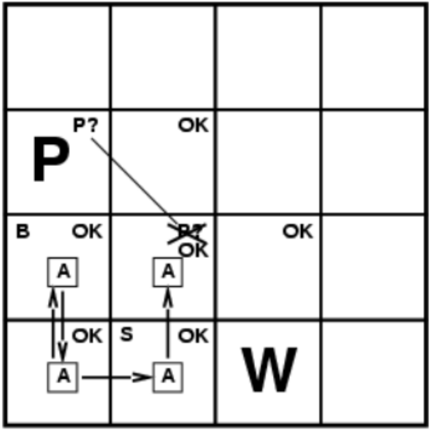
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world

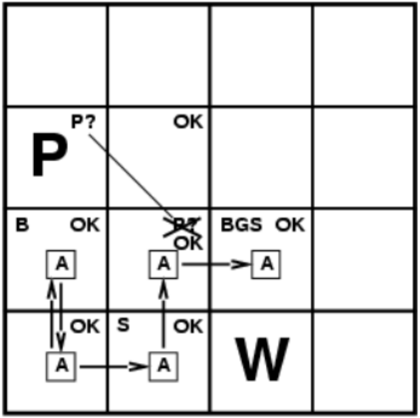


Table of Contents

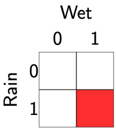
Ingredients of a logic

- Logics are formal languages for representing information such that conclusions can be draw
- Syntax: defines a set of valid **formulas** (Formulas)

  Example: $Rain \wedge Wet$

- Semantics: for each formula, specify a set of models (assignments / configurations of the world)

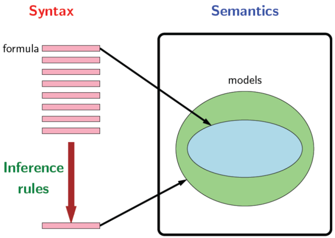
  Example:



- Inference rules: given f, what new formulas g can be added that are guaranteed to follow (like, Modus Ponens (MP))?

  Example: from $Rain \wedge Wet$*, derive* $Rain$

Schema for logic



- Entailment: $KB \models \alpha$ iff α is true in all worlds where KB is true iff $\mathcal{M}(KB) \subseteq \mathcal{M}(\alpha)$
- Inference: $KB \vdash \alpha$ iff α can be inferred (or derived, or deduced) from KB by a **procedure**
- Soundness:

  whenever $KB \vdash \alpha$, *it is also true that* $KB \models \alpha$

- Completeness:

  whenever $KB \models \alpha$, *it is also true that* $KB \vdash \alpha$

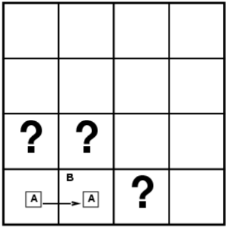
Entailment in the wumpus world

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]—知识库KB

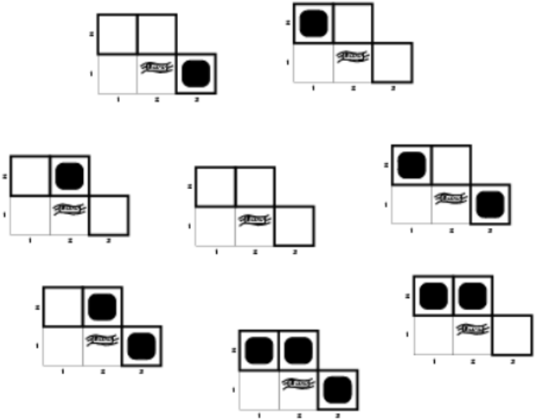
Consider possible models for *KB* assuming only pits

考虑相邻的方格是否包含陷阱

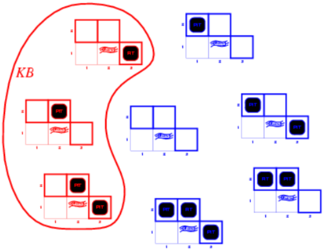
3 Boolean choices ⇒ 8 possible models



Wumpus models

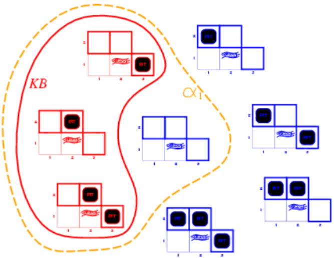


Wumpus models



KB = wumpus-world rules + observations

Wumpus models

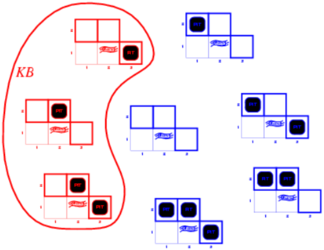


KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking（模型检验）

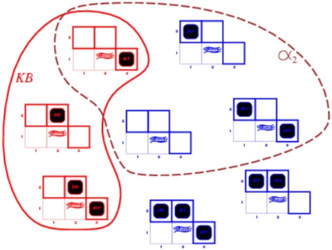
- 在KB为真的每个模型中，α_1 也为真，因此$KB \models \alpha_1$

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

- 在KB为真的某些模型中，α_2 为假，因此$KB \not\models \alpha_2$

Logics

- **Propositional logic with only Horn clauses**
- **Propositional logic**
- Modal logic
- **First-order logic with only Horn clauses**
- **First-order logic**
- Second-order logic
- Non-monotonic logic: Default logic, Autoepistemic logic, Circumscription, MKNF (MBNF)
- ...

Key idea: tradeoff

Balance **expressivity** and **computational efficiency**.

Table of Contents

Syntax of propositional logic

- Propositional symbols (atomic formulas): A, B, C, ...
- Logical connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- Build up formulas recursively–if f and g are formulas, so are the following:
  - Negation: $\neg f$
  - Conjunction: $f \wedge g$
  - Disjunction: $f \vee g$
  - Implication: $f \rightarrow g$
  - Biconditional: $f \leftrightarrow g$
- Formulas by themselves are just symbols (syntax). No meaning yet (semantics)!
- **Atom**: atomic formula
- **Literal**: atomic formula or negated atomic formula
- **Clause**: disjunction of literals

Model

定义 (Model)

A **model** *w* in propositional logic is an **assignment** of truth values to propositional symbols.

Example:

- 3 propositional symbols: A, B, C
- $2^3 = 8$ possible models *w*:

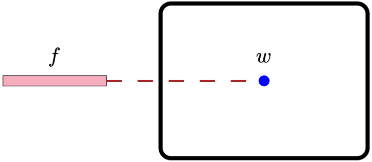
$$\{A:0, B:0, C:0\}$$
$$\{A:0, B:0, C:1\}$$
$$\{A:0, B:1, C:0\}$$
$$\{A:0, B:1, C:1\}$$
$$\{A:1, B:0, C:0\}$$
$$\{A:1, B:0, C:1\}$$
$$\{A:1, B:1, C:0\}$$
$$\{A:1, B:1, C:1\}$$

Interpretation function

定义 (Interpretation function)

Let f be a formula and w a model. An **interpretation function** $\mathcal{I}(f, w)$ returns:

- true (1) (say that w satisfies f)
- false (0) (say that w does not satisfies f)



Interpretation function: definition

- Base case:
  - For a propositional symbol p (e.g., A, B, C): $\mathcal{I}(p, w) = w(p)$
- Recursive case:
  - For any two formulas f and g, define:

$\mathcal{I}(f,w)$	$\mathcal{I}(g,w)$	$\mathcal{I}(\neg f,w)$	$\mathcal{I}(f \wedge g,w)$	$\mathcal{I}(f \vee g,w)$	$\mathcal{I}(f \to g,w)$	$\mathcal{I}(f \leftrightarrow g,w)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Interpretation function: example

Example: interpretation function

Formula: $f = (\neg A \wedge B) \leftrightarrow C$

Model: $w = \{A : 1, B : 1, C : 0\}$

Interpretation:

- $\mathcal{I}((\neg A \wedge B) \leftrightarrow C, w) = 1$
  - $\mathcal{I}(\neg A \wedge B, w) = 0$
    - $\mathcal{I}(\neg A, w) = 0$
      - $\mathcal{I}(A, w) = 1$
    - $\mathcal{I}(B, w) = 1$
  - $\mathcal{I}(C, w) = 0$

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j] .

Let $B_{i,j}$ be true if there is a breeze in [i, j] .

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j] .

Let $B_{i,j}$ be true if there is a breeze in [i, j] .

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

"A square is breezy if and only if there is an adjacent pit"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

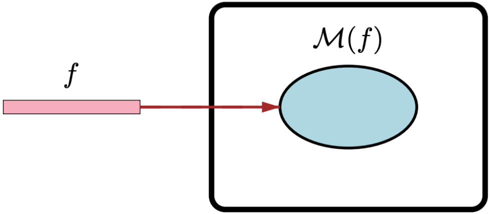
Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Formula represents a set of models

So far: each formula f and model w has an interpretation $\mathcal{I}(f, w) \in \{0, 1\}$

定义 (Models)

Let $\mathcal{M}(f)$ be the set of **models** w for which $\mathcal{I}(f, w) = 1$.



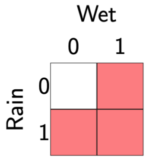
Models: example

Formula:

$$f = \text{Rain} \vee \text{Wet}$$

Models:

$$\mathcal{M}(f) =$$



Key idea: compact representation

A **formula** *compactly* represents a set of **models**.

Knowledge base

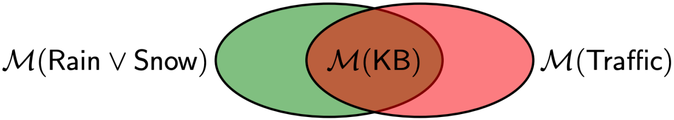
定义 (Knowledge base)

A **knowledge base** KB is a set of formulas representing their conjunction / intersection:

$$\mathcal{M}(KB) = \bigcap_{f \in KB} \mathcal{M}(f).$$

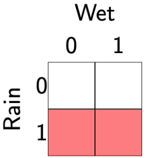
Intuition: KB specifies constraints on the world. $\mathcal{M}(KB)$ is the set of all worlds satisfying those constraints.

Let $\text{KB} = \{\text{Rain} \vee \text{Snow}, \text{Traffic}\}$.

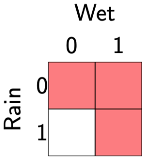


Knowledge base: example

$\mathcal{M}(\text{Rain})$

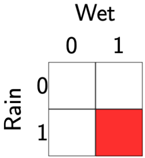


$\mathcal{M}(\text{Rain} \rightarrow \text{Wet})$



Intersection:

$\mathcal{M}(\{\text{Rain}, \text{Rain} \rightarrow \text{Wet}\})$

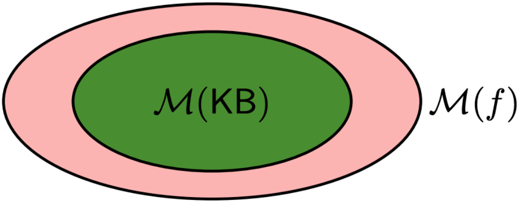


Entailment

定义

Entailment KB entails f (written $KB \models f$) iff $\mathcal{M}(f) \supseteq \mathcal{M}(KB)$.

- Intuition: f added no information/constraints (it was already known)
- Example: $\text{Rain} \wedge \text{Snow} \models \text{Snow}$

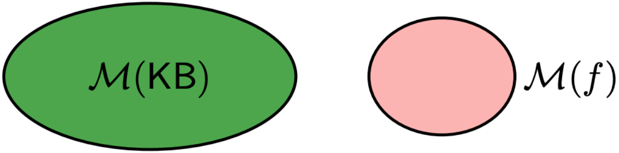


Contradiction

定义

Contradiction KB contradicts f iff $\mathcal{M}(KB) \cap \mathcal{M}(f) = \emptyset$.

- Intuition: f contradicts what we know (captured in KB)
- Example: Rain $\wedge$ Snow contradicts $\neg$Snow

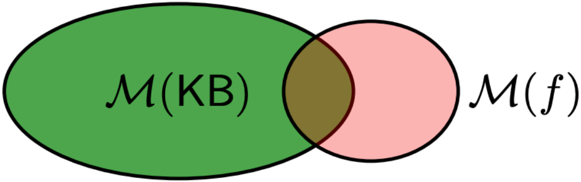


Contingency

- Intuition: f adds non-trivial information to KB

$$\emptyset \subset \mathcal{M}(KB) \cap \mathcal{M}(f) \subset \mathcal{M}(KB).$$

- Example: Rain and Snow

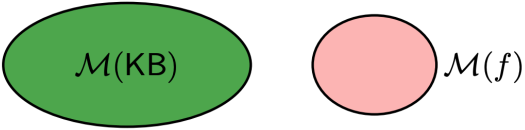


Contradiction and entailment

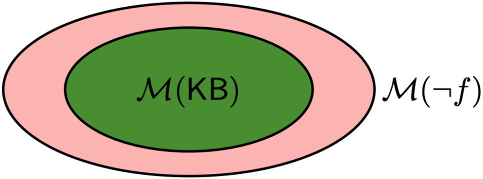
Proposition (Contradiction and entailment)

KB contradicts f iff KB entails $\neg f$*.*

Contradiction:



Entailment:



Tell operation

$$\text{Tell}[f] \longrightarrow \boxed{\text{KB}} \longrightarrow \ ?$$

Tell: *It is raining.*

$$\text{Tell}[\text{Rain}]$$

Possible responses:

- Already knew that: entailment ($\text{KB} \models f$)
- Don't believe that: contradiction ($\text{KB} \models \neg f$)
- Learned something new (update KB): contingent

Ask operation

$$\text{Ask}[f] \longrightarrow \boxed{\text{KB}} \longrightarrow \;?$$

Ask: *Is it raining?*

$$\text{Ask}[\text{Rain}]$$

Possible responses:

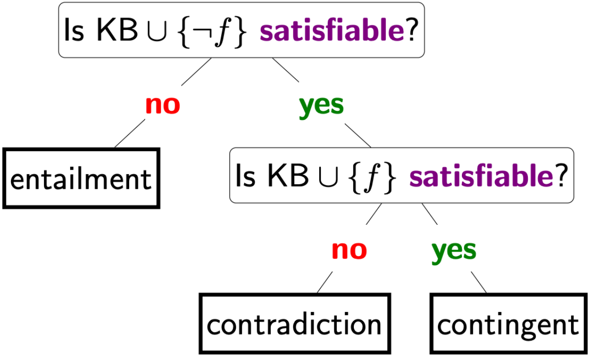
- Yes: entailment ($\text{KB} \models f$)
- No: contradiction ($\text{KB} \models \neg f$)
- I don't know: contingent

Satisfiability

定义 (Satisfiability)

A knowledge base KB is satisfiable if $\mathcal{M}(KB) \neq \emptyset$.

Reduce Ask$[f]$ and Tell$[f]$ to satisfiability:



Model checking

- Checking satisfiability (SAT) in propositional logic is special case of solving CSPs!
- Mapping:

propositional symbol	$\Rightarrow$	variable
formula	$\Rightarrow$	constraint
model	$\Leftarrow$	assignment

Model checking

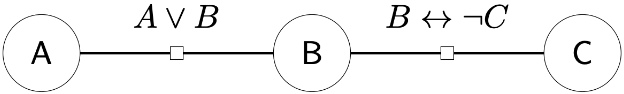
Example: model checking

KB $= \{A \vee B, B \leftrightarrow \neg C\}$

Propositional symbols (CSP variables):

$$\{A, B, C\}$$

CSP:



Consistent assignment (satisfying model):

$$\{A : 1, B : 0, C : 1\}$$

Model checking

定义 (Model checking)

Input: knowledge base KB
Output: exists satisfying model ($\mathcal{M}(KB) \neq \emptyset$)?

- Popular algorithms:
  - DPLL (backtracking search + pruning)
  - WalkSat (randomized local search)
- Next: Can we exploit the fact that factors are formulas?

Inference rules

Example of making an inference:

It is raining. (Rain)

If it is raining, then it is wet. (Rain $\rightarrow$ Wet)

Therefore, it is wet. (Wet)

$$\frac{\text{Rain}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Wet}} \qquad \frac{\text{(premises)}}{\text{(conclusion)}}$$

定义 (Modus ponens inference rule)

For any propositional symbols p and q:

$$\frac{p, \quad p \rightarrow q}{q}$$

Inference framework

定义

If $f_1, \ldots, f_k, g$ are formulas, then the following is an inference rule:

$$\frac{f_1, \ \ldots, \ f_k}{g}$$

Key idea: inference rules

Rules operate directly on **syntax**, not on **semantics**.

Inference algorithm

Algorithm: forward inference

Input: set of inference rules Rules.

Repeat until no changes to KB:

  Choose set of formulas $f_1, \ldots, f_k \in$ KB.

  If matching rule $\frac{f_1, \quad \ldots \quad , f_k}{g}$ exists:

    Add g to KB.

定义 (Derivation)

KB derives/proves f ($KB \vdash f$) iff f eventually gets added to *KB*.

Inference example

Example: Modus ponens inference

Starting point:

$$\text{KB} = \{\text{Rain}, \text{Rain} \to \text{Wet}, \text{Wet} \to \text{Slippery}\}$$

Apply modus ponens to Rain and Rain $\to$ Wet:

$$\text{KB} = \{\text{Rain}, \text{Rain} \to \text{Wet}, \text{Wet} \to \text{Slippery}, \text{Wet}\}$$

Apply modus ponens to Wet and Wet $\to$ Slippery:

$$\text{KB} = \{\text{Rain}, \text{Rain} \to \text{Wet}, \text{Wet} \to \text{Slippery}, \text{Wet}, \text{Slippery}\}$$

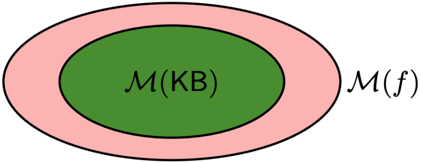
Converged.

Can't derive some formulas: $\neg$Wet, Rain $\to$ Slippery

Desidarata for inference rules

Semantics

Interpretation defines **entailed/true** formulas: KB $\models f$:



Syntax:

Inference rules **derive** formulas: KB $\vdash f$

How does $\{f : \text{KB} \models f\}$ relate to $\{f : \text{KB} \vdash f\}$?

Soundness and Completeness

定义 (Soundness)

A set of inference rules Rules is sound if:

$$\{f: KB \vdash f\} \subseteq \{f: KB \models f\}.$$

定义 (Completeness)

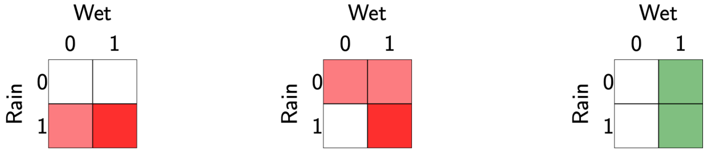
A set of inference rules Rules is complete if:

$$\{f: KB \vdash f\} \supseteq \{f: KB \models f\}.$$

Soundness: example

Is $\dfrac{\text{Rain}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Wet}}$ (Modus ponens) sound?

$$\mathcal{M}(\text{Rain}) \quad \cap \quad \mathcal{M}(\text{Rain} \rightarrow \text{Wet}) \quad \subseteq ? \quad \mathcal{M}(\text{Wet})$$

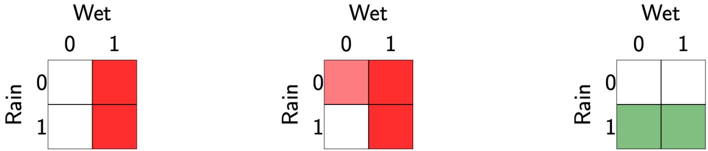


Sound!

Soundness: example

Is $\dfrac{\text{Wet}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Rain}}$ sound?

$$\mathcal{M}(\text{Wet}) \quad \cap \quad \mathcal{M}(\text{Rain} \rightarrow \text{Wet}) \quad \subseteq? \quad \mathcal{M}(\text{Rain})$$



Unsound!

Completeness: example

Recall completeness: inference rules derive all entailed formulas (f such that $KB \models f$)

Example: Modus ponens is incomplete

Setup:

$$\text{KB} = \{\text{Rain}, \text{Rain} \vee \text{Snow} \rightarrow \text{Wet}\}$$

$$f = \text{Wet}$$

$$\text{Rules} = \{\frac{f, \quad f \rightarrow g}{g}\} \text{ (Modus ponens)}$$

Semantically: $\text{KB} \models f$ (f is entailed).

Syntactically: $\text{KB} \not\vdash f$ (can't derive f).

Incomplete!

Fixing completeness

Option 1: Restrict the allowed set of formulas

propositional logic

↓

propositional logic with only Horn clauses

Option 2: Use more powerful inference rules

Modus ponens

↓

resolution

Table of Contents

Proof methods

Proof methods divide into (roughly) two kinds:

- Application of inference rules 推理规则的应用
  - Legitimate (sound) generation of new sentences from old
  - Proof = a sequence of inference rule applications 推理规则的应用序列
  - Can use inference rules as operators in a standard search alg.
  - 寻找证明的过程与搜索问题中寻找解的过程非常类似：定义后继函数以便生成推理规则所有可能的应用。
  - Typically require translation of sentences into a normal form
- Model checking
  - Truth table enumeration (always exponential in n)
  - Improved backtracking, e.g., Davis-Putnam-Logemann-Loveland
  - Heuristic search in model space (sound but incomplete), e.g., min-conflicts-like hill-climbing algorithms

Definite clauses

定义 (Definite clauses)

A **definite clause** has the following form:

$$(p_1 \wedge \cdots \wedge p_k) \to q$$

where $p_1, \ldots, p_k$ $(k \geq 0)$ and q are propositional symbols.

Intuition: If $p_1, \ldots, p_k$ hold, then q holds.

Examples:

- $(\text{Rain} \wedge \text{Snow}) \to \text{Traffic}$
- Traffic

Non-examples:

- $\text{Rain} \wedge \text{Snow}$
- $\neg\text{Traffic}$
- $(\text{Rain} \wedge \text{Snow}) \to (\text{Traffic} \vee \text{Peaceful})$

Horn clauses

定义 (Horn clauses)

A **Horn clause** is either:

- a definite clause $(p_1 \wedge \cdots \wedge p_k \rightarrow q)$
- a goal clause $(p_1 \wedge \cdots \wedge p_k) \rightarrow \bot$

Examples:

- Definite:

$$(\text{Rain} \wedge \text{Snow}) \rightarrow \text{Traffic}$$

- Goal:

$$(\text{Traffic} \wedge \text{Accident}) \rightarrow \bot$$

  equivalent:

$$\neg(\text{Traffic} \wedge \text{Accident})$$

Modus ponens

定义 (Modus ponens)

$$\frac{p_1, \ \ldots, \ p_k, \quad (p_1, \ldots, p_k) \to q}{q}$$

Example:

Example: Modus ponens

$$\frac{\text{Wet}, \quad \text{Weekday}, \quad \text{Wet} \land \text{Weekday} \to \text{Traffic}}{\text{Traffic}}$$

Can be used with forward chaining or backward chaining.
These algorithms are very natural and run in linear time.

Completeness of modus ponens

定理 (Modus ponens on Horn clauses)

Modus ponens is complete with respect to Horn clauses:

- *Suppose KB contains only Horn clauses and p is an entailed propositional symbol.*
- *Then applying modus ponens will derive p.*

Upshot:

$KB \models p$ *(entailment) is the same as* $KB \vdash p$ *(derivation)!*

Answering questions

KB

Rain

Weekday

Rain $\rightarrow$ Wet

Wet $\wedge$ Weekday $\rightarrow$ Traffic

Traffic $\wedge$ Careless $\rightarrow$ Accident

Definition: Modus ponens

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

Query: Ask[Traffic]

"Yes" subproblem: KB $\models$ Traffic

Equivalent: KB contradicts $\neg$Traffic

Equivalent: KB $\cup$ {Traffic $\rightarrow$ false} $\vdash$ false?

"No" subproblem: KB $\models$ $\neg$Traffic

Equivalent: KB $\vdash$ $\neg$Traffic — **impossible!**

Answering questions

- Note that for the "no" subproblem, it is actually impossible to derive $\neg$Traffic. This is because modus ponens can only generate propositional symbols, not their negations. This means always either answer "yes" or "I don't know".
- This is reasonable because setting all variables to true is always a valid model no matter what the Horn clauses are, so we can never say "no".

"Yes" subproblem

KB

Rain

Weekday

Rain → Wet

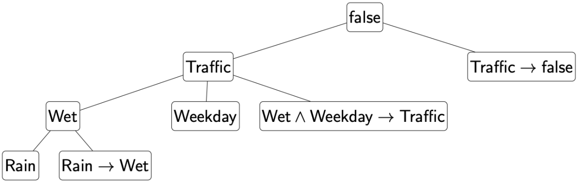
Wet ∧ Weekday → Traffic

Traffic ∧ Careless → Accident

Definition: Modus ponens

$$\frac{p_1, \quad \cdots \quad, p_k, \quad (p_1 \land \cdots \land p_k) \to q}{q}$$

Question: KB ∪ {Traffic → false} ⊢ false?

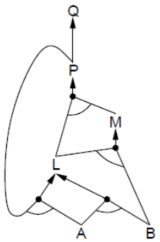


Forward chaining

Idea: fire any rule whose premises （前提） are satisfied in the *KB*, add its conclusion to the *KB*, until query （询问） is found

从知识库中的已知事实（正文字）开始。如果蕴涵的所有前提已知，那么把它的结论加到已知事实集。持续这一过程，直到询问q被添加或者直到无法进行更进一步的推理

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

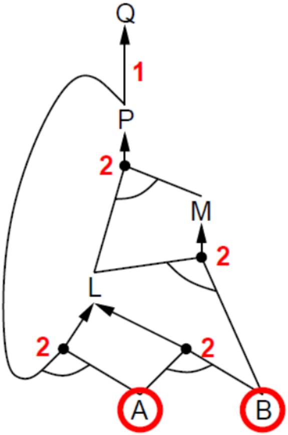


Forward chaining algorithm

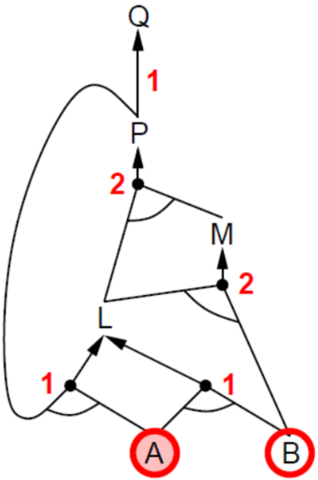
```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known in KB

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← true
           for each Horn clause c in whose premise p appears do
               decrement count[c]
               if count[c] = 0 then do
                   if HEAD[c] = q then return true
                   PUSH(HEAD[c], agenda)
   return false
```

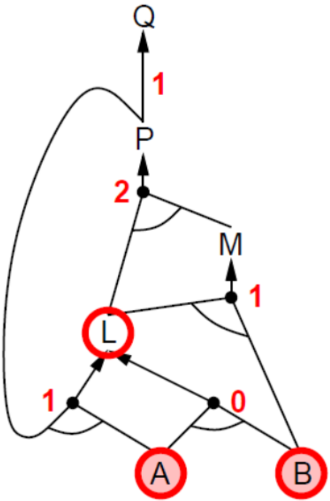
Forward chaining example



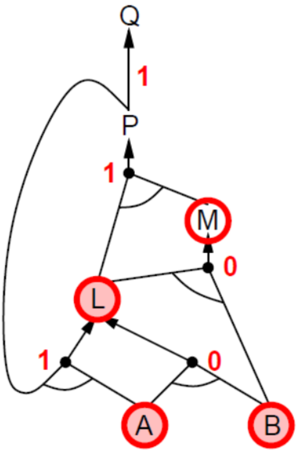
Forward chaining example



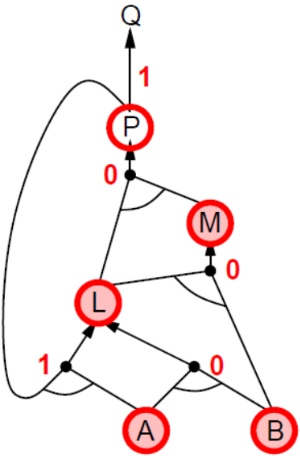
Forward chaining example



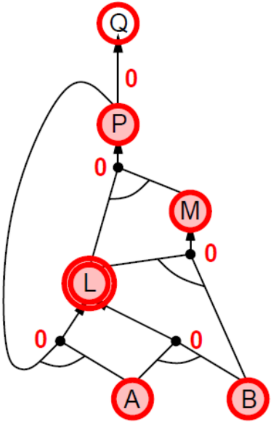
Forward chaining example



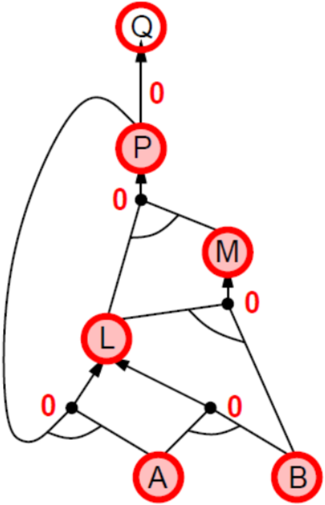
Forward chaining example



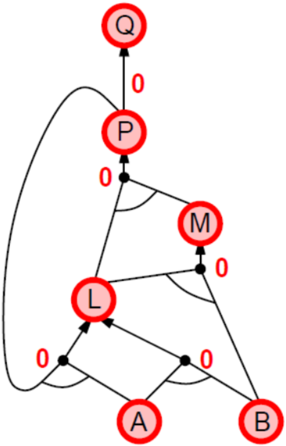
Forward chaining example



Forward chaining example



Forward chaining example



Properties of forward chaining

- 对于对于 Horn KB，Forward chaining 是
  - 可靠的：每个推理本质上是分离规则的一个应用
  - 完备的：每个被蕴含的原子语句都将得以生成

Proof of completeness

FC可推出每个被KB蕴涵的**原子**语句

1. FC到达不动点以后，不可能再出现新的推理。
2. 考察inferred表的最终状态，参与推理过程的每个符号为true，其它为false。把该推理表看做一个逻辑模型*m*
3. *原始KB中的每个确定子句在该模型m中都为真*

   证明：假设某个子句 $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ 在*m*中为false

   那么 $a_1 \wedge \ldots \wedge a_k$ 在*m*中为true， b 在*m*中为false

   与算法已经到达一个不动点相矛盾
4. m是*KB*的一个模型
5. 如果 $KB \models q$ ，q在*KB*的所有模型中必须为真，包括*m*
6. q在*m*中为真→在inferred表中为真→被FC算法推断出来

Backward chaining

Idea: 从查询q反向进行:

to prove q by BC,

check if q is known already（检查是否q已知为真）, or

prove by BC all premises of some rule concluding q

（寻找知识库中那些以q为结论的蕴涵，证明其中一个蕴涵的所有前提为真）

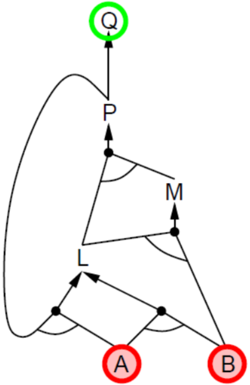
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

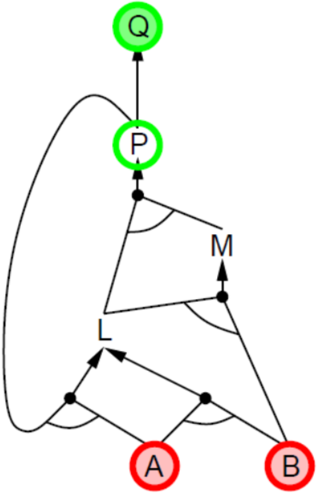
1) has already been proved true, or

2) has already failed

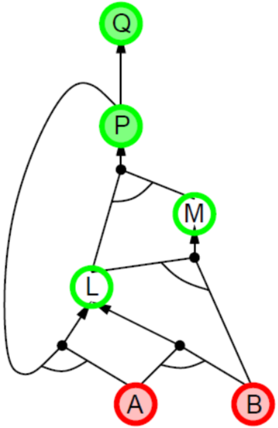
Backward chaining example



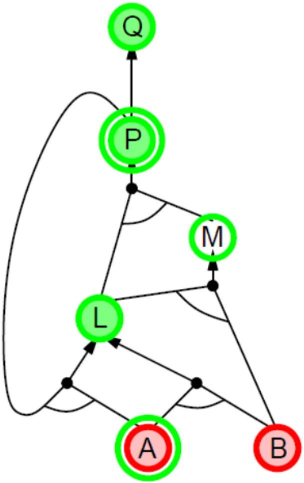
Backward chaining example



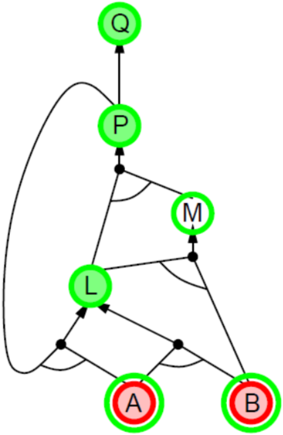
Backward chaining example



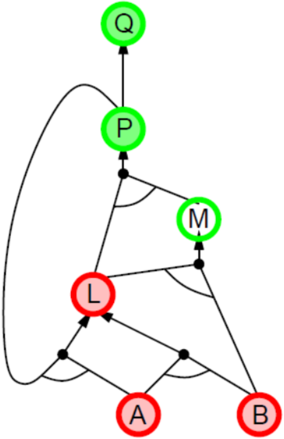
Backward chaining example



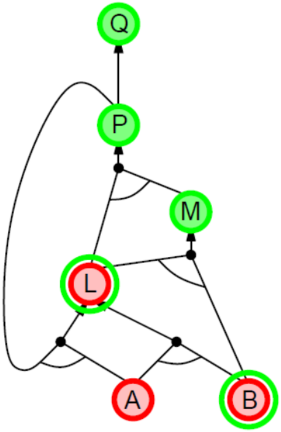
Backward chaining example



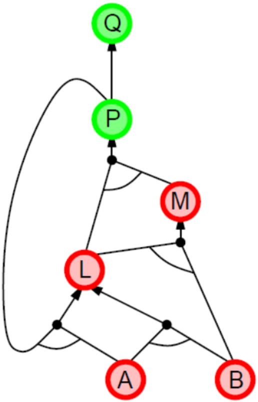
Backward chaining example



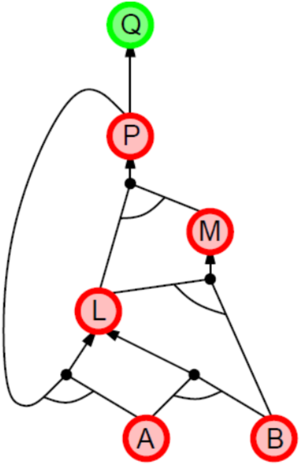
Backward chaining example



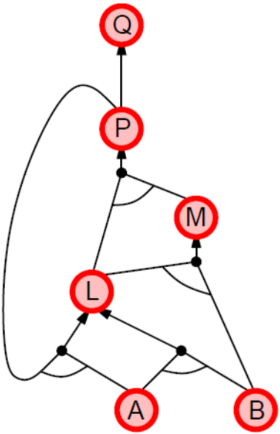
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

FC is data-driven（数据驱动）, cf. automatic, unconscious processing,
e.g., object recognition, routine decisions

May do lots of work that is irrelevant to the goal

BC is goal-driven（目标指导）, appropriate for problem-solving,
e.g., Where are my keys? How do I get into a PhD program?

Complexity of BC can be much less than linear in size of KB

Resolution 归结

Conjunctive Normal Form 合取范式 (CNF)

conjunction of disjunctions of literals（文字析取式的合取式）

(disjunctions of literals = clauses)

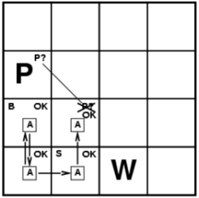
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Resolution inference rule归结推理规则 (for CNF):

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

where ℓ_i and m_j are complementary literals（互补文字）

E.g., $$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is sound and complete for propositional logic
命题逻辑中归结是可靠和完备的

Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution algorithm

- Recall: KB operation boil down to satisfiability

  $KB \models \alpha$ if and only if ($KB \wedge \neg\alpha$) is unsatisfiable

- Algorithm: resolution-based inference
  - Convert all formulas to CNF
  - Repeatedly apply resolution rule
  - Return unsatisfaible iff derive false

Resolution algorithm

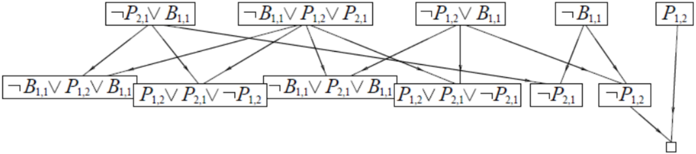
Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$ $\alpha = \neg P_{1,2}$



Time complexity

- Modus ponens inference rule

$$\frac{p_1, \quad \cdots \quad, p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

  - Each rule application adds clause with **one** propositional symbol → linear time

- Resolution inference rule

$$\frac{f_1 \vee \cdots \vee f_n \vee h, \quad \neg h \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

  - Each rule application adds clause with **many** propositional symbols → exponential time

Comparison

Horn clauses	any clauses
Modus ponens	resolution
linear time	exponential time
less expressive	more expressive

Summary

Logical agents apply inference to a knowledge base
to derive new information and make decisions

Basic concepts of logic:

— **syntax: formal structure of sentences**
— **semantics: truth of sentences wrt models**
— **entailment: necessary truth of one sentence given another**
— **inference: deriving sentences from other sentences**
— **soundness: derivations produce only entailed sentences**
— **completeness: derivations can produce all entailed sentences**

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Forward, backward chaining are linear-time, complete for Horn clauses
Resolution is complete for propositional logic

Propositional logic lacks expressive power

Homework

- 7.13（第三版）
- Prove the completeness of the forward chaining algorithm