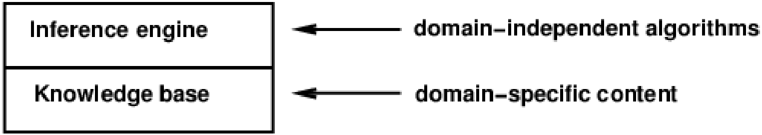
Lecture 7: Logical Agents and Propositional Logic

CS 580 (001) - Spring 2018

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

March 07, 2018



Knowledge base = set of sentences in a **formal** language
Declarative approach to building an agent (or other system):
  TELL it what it needs to know
  Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level
  i.e., **what they know**, regardless of how implemented

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

```
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.
Incorporate new percepts
Update internal representations of the world
Deduce hidden properties of the world
Deduce appropriate actions

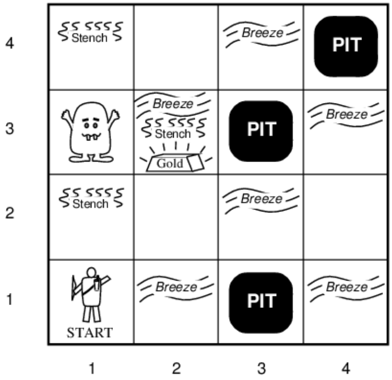
Wumpus World - PEAS Description

Performance measure
- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square

Actuators Left turn, Right turn,
- Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell



Observable??

Observable?? No—only local perception

Observable?? No—only local perception

Deterministic??

Observable?? No—only local perception

Deterministic?? Yes—outcomes exactly specified

Observable?? No—only local perception

Deterministic?? Yes—outcomes exactly specified

Episodic??

Observable?? No—only local perception

Deterministic?? Yes—outcomes exactly specified

Episodic?? No—sequential at the level of actions

Wumpus World Characterization

Observable?? No—only local perception

Deterministic?? Yes—outcomes exactly specified

Episodic?? No—sequential at the level of actions

Static??

Observable?? 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

Observable?? 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??

Wumpus World Characterization

Observable?? 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

Wumpus World Characterization

Observable?? 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??

Wumpus World Characterization

Observable?? 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

Wumpus World Characterization

Observable?? No—only local perception

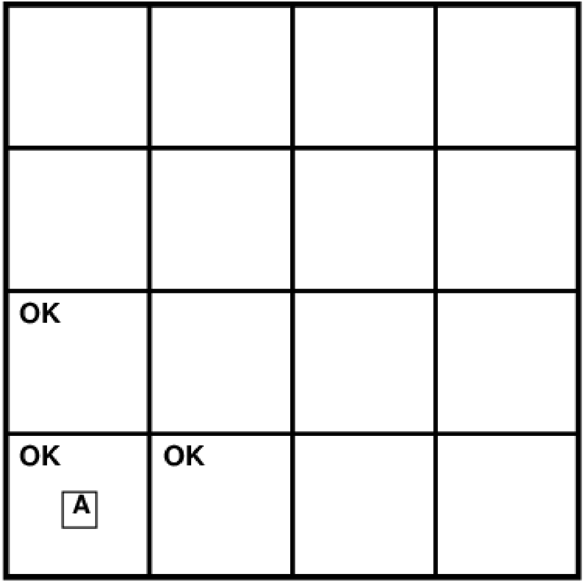
Deterministic?? Yes—outcomes exactly specified

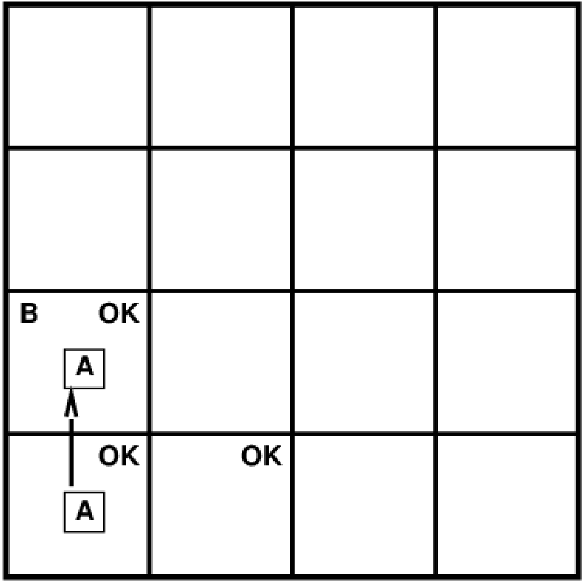
Episodic?? No—sequential at the level of actions

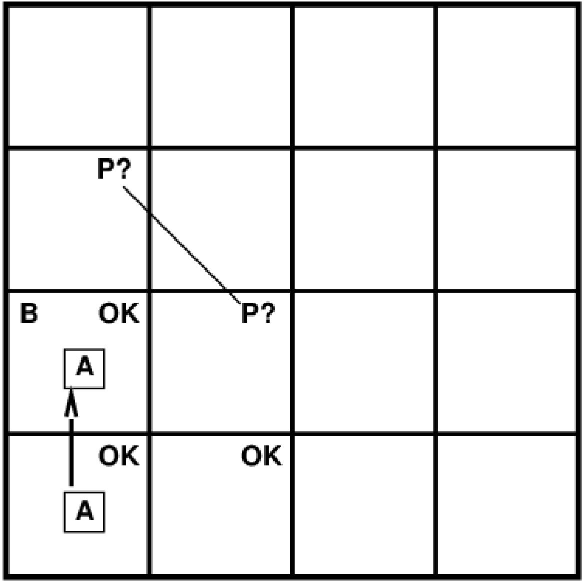
Static?? Yes—Wumpus and Pits do not move

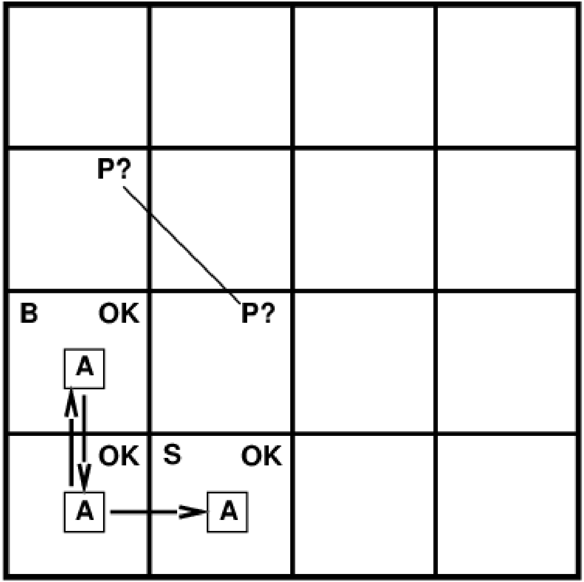
Discrete?? Yes

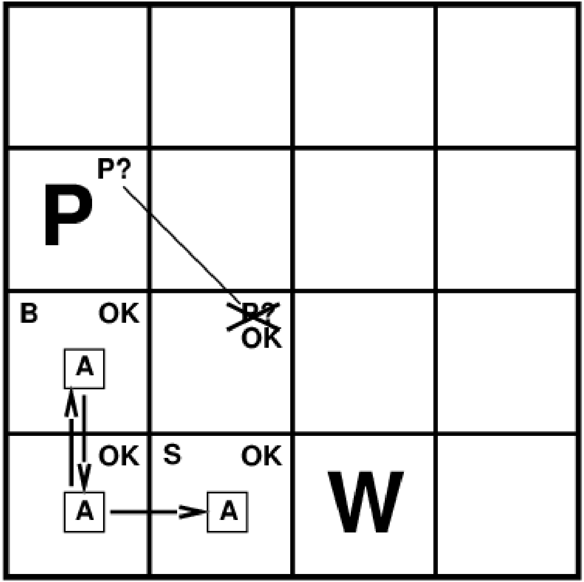
Single-agent?? Yes—Wumpus is essentially a natural feature


OK
OK
A
OK

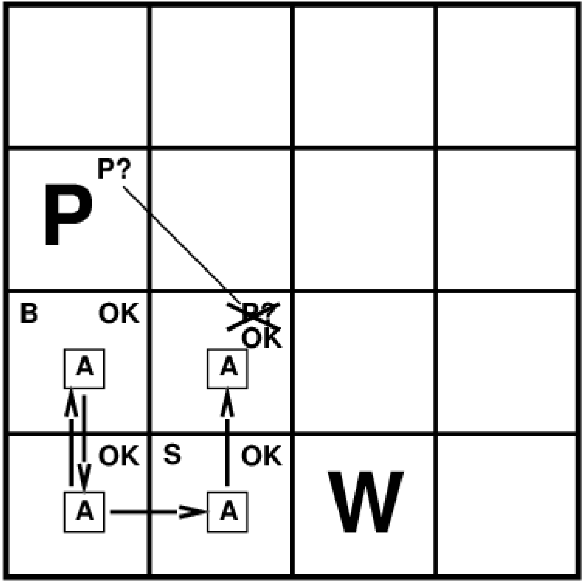

B
OK
A
OK
OK
A


P?
B
OK
P?
A
OK
OK
A

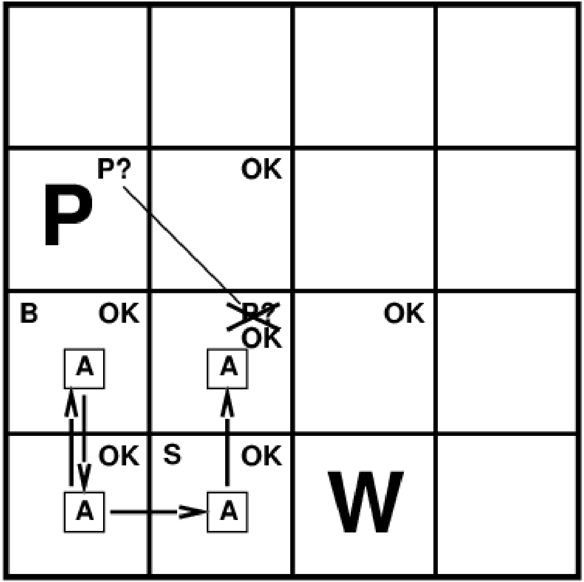

P?
B
OK
P?
A
OK
S
OK
A
A


P?
P
B
OK
~~P?~~
OK
A
OK
S
OK
W
A
A

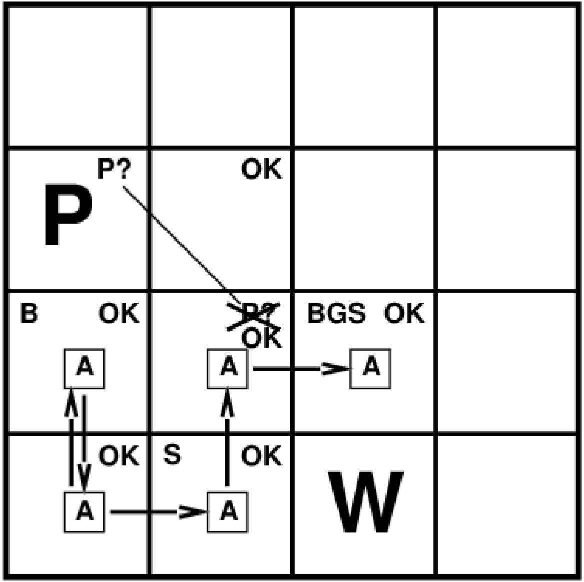
Exploring a Wumpus World

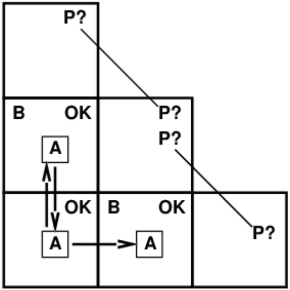


Exploring a Wumpus World



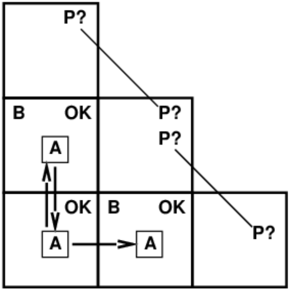
Exploring a Wumpus World





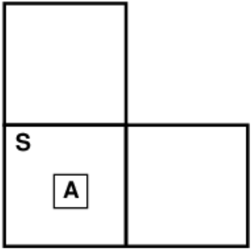
Breeze in (1,2) and (2,1)
$\Longrightarrow$ no safe actions

Assuming pits uniformly distributed,
(2,2) has pit w/ higher probability (how much?)



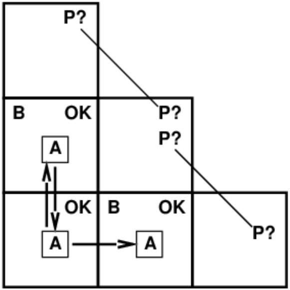
Breeze in (1,2) and (2,1)
$\implies$ no safe actions

Assuming pits uniformly distributed,
(2,2) has pit w/ higher probability (how much?)



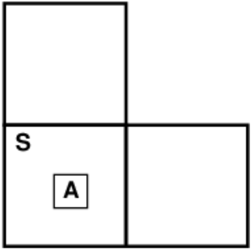
Smell in (1,1)
$\implies$ cannot move

Can use a strategy of coercion:
shoot straight ahead
wumpus was there $\implies$ dead $\implies$ safe
wumpus wasn't there $\implies$ safe



Breeze in (1,2) and (2,1)
$\Longrightarrow$ no safe actions

Assuming pits uniformly distributed,
(2,2) has pit w/ higher probability (how much?)



Smell in (1,1)
$\Longrightarrow$ cannot move

Can use a strategy of coercion:
shoot straight ahead
wumpus was there $\Longrightarrow$ dead $\Longrightarrow$ safe
wumpus wasn't there $\Longrightarrow$ safe

Logic in General

Logics are formal languages for representing information
such that conclusions can be drawn

Syntax determines how sentences are expressed in a particular logic/language

Semantics define the "meaning" of sentences;
i.e., define truth of a sentence in a world

E.g., the language of arithmetic

$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence
$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
$x + 2 \geq y$ is true in a world where $x = 7,\ y = 1$
$x + 2 \geq y$ is false in a world where $x = 0,\ y = 6$

Types of Logic

Logics are characterized by what they commit to as primitives
Ontological commitment: what exists—facts? objects? time? beliefs?
Epistemological commitment: what states of knowledge?

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Reasoning with Logic

- First order of business: fundamental concepts of logical representation and reasoning

  independent of any logic's particular form/type

  Entailment

- Second order of business: Introduction to **propositional logic**

  Wumpus KB via propositional logic

- Third order of business: Drawing conclusions

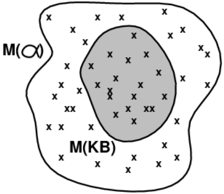
  Inference and theorem proving

- Can use the term **model** in place of possible world
- Logicians typically think in terms of **models**, which are formally-structured worlds with respect to which truth can be evaluated
- Model = mathematical abstraction that fixes the truth/falsehood of every relevant sentence
- Possible models are just all possible assignments of variables in the environment
- We say that a model m "satisfies" sentence α if α "is true in" m
  Or: "m is a model of α"
  $M(\alpha)$ is the set of all models of α

Entailment means that one thing **follows from** another:

$$KB \models \alpha$$

Knowledge base KB entails sentence α
iff α is true in all worlds/models where KB is true
$KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$



E.g., KB containing "Giants won" and "Reds won" entails "Giants or Reds won"
$x + y = 4$ entails $4 = x + y$

Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

Note: brains process **syntax** (of some sort)

Quick Exercise

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

$M(\alpha) \subseteq M(\beta)$

Quick Exercise

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

$M(\alpha) \subseteq M(\beta)$

β is satisfied in all models of α

Quick Exercise

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

$M(\alpha) \subseteq M(\beta)$

β is satisfied in all models of α

β may be satisfied in other models, as well

Quick Exercise

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

$M(\alpha) \subseteq M(\beta)$

β is satisfied in all models of α

β may be satisfied in other models, as well

α is a *stronger* assertion than β

Quick Exercise

Given two sentences α and β, what does this mean:

$$\alpha \models \beta$$

α entails β

$M(\alpha) \subseteq M(\beta)$

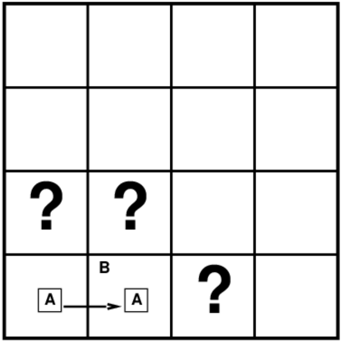
β is satisfied in all models of α

β may be satisfied in other models, as well

α is a *stronger* assertion than β

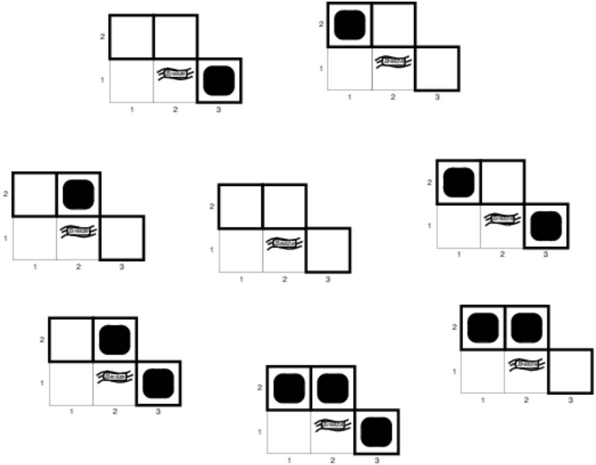
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

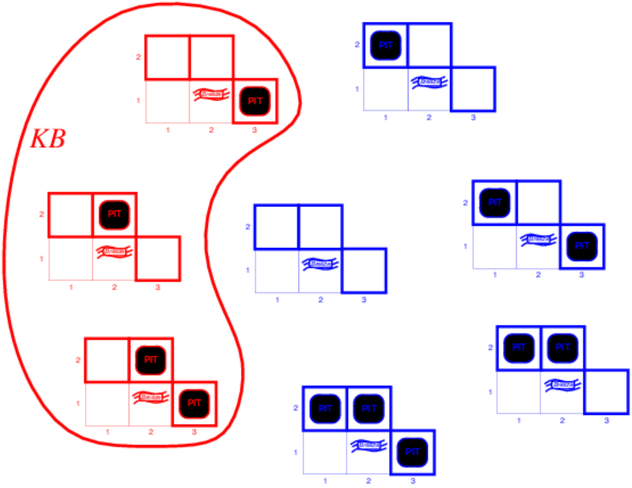
Consider possible models for ?s assuming only pits



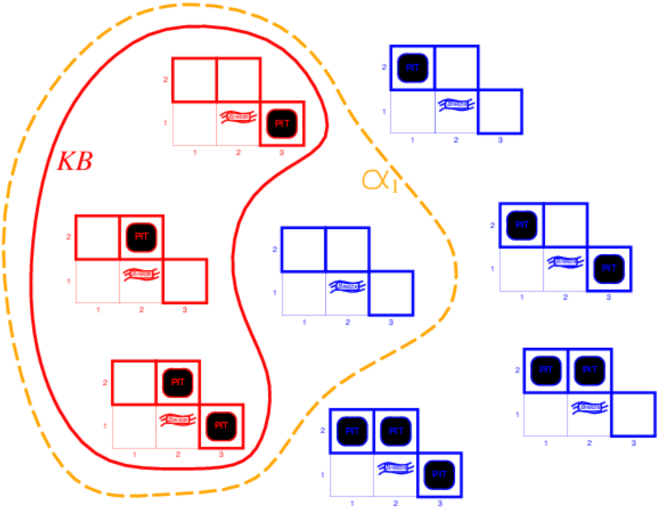
3 Boolean choices $\Longrightarrow$ 8 possible models

Wumpus Models



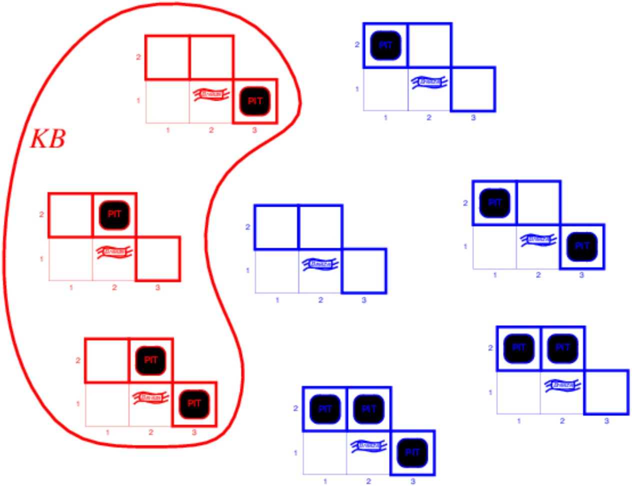


KB = wumpus-world rules + observations

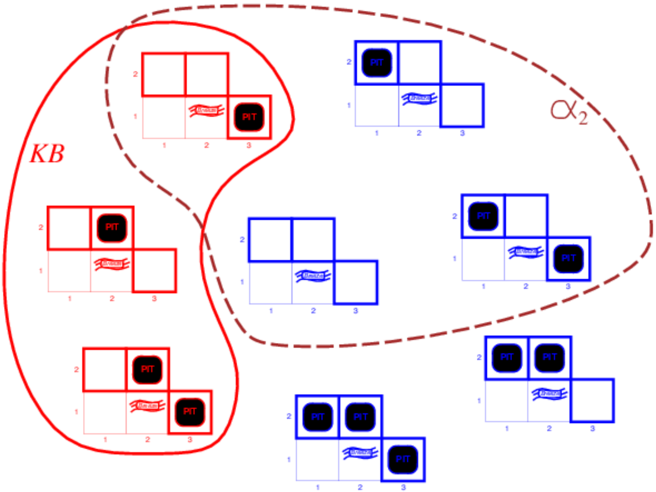


KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking



KB = wumpus-world rules + observations



KB = wumpus-world rules + observations

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

From Entailment to Logical Inference

Entailment can be used to derive logical conclusions

i.e.: carry out logical inference

A straightforward algorithm to carry out inference:

Model checking

Model checking enumerates all possible models to check that α is true in all models where KB is true

i.e.: $M(KB) \subseteq M(\alpha)$

To understand entailment and inference: haystack and needle analogy

Consequences of KB are a haystack; α is a needle.

Entailment = needle in haystack

inference = finding it

We need inference procedures to derive α from a given KB

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Right now, we will venture into propositional logic; first-order logic is next.

$KB \vdash_i \alpha =$ sentence α can be derived from KB by procedure i

Soundness: inference procedure i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

Completeness: inference procedure i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Right now, we will venture into propositional logic; first-order logic is next.

Propositional Logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

Atomic sentences consist of a single proposition symbol
E.g.: Proposition symbols P_1, P_2, etc. are atomic sentences

Each such symbol stands for a proposition that can be true or false
E.g.: $W_{1,3}$ stands for proposition that wumpus is in [1,3]

Two propositions with fixed meaning: *True* and *False*

Complex sentences built over atomic ones via connectives:
negation, conjunction, disjunction, implication, biconditional

If S is a sentence, $\neg S$ is a sentence (negation)
  A (positive) literal is an atomic sentence
  A (negative) literal is a negated atomic sentence

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  S_1 and S_2 are called conjuncts

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  S_1 and S_2 are called disjuncts

If S_1 and S_2 are sentences, $S_1 \implies S_2$ is a sentence (implication/conditional)
  S_1 is called premise/antecedent
  S_2 is called conclusion or consequent

  implication also known as rule or if-then statement

If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

BNF is an ambiguous formal grammar for propositional logic (pg. 1060 if unfamiliar):

Sentence $\rightarrow$ AtomicSentence | ComplexSentence

AtomicSentence $\rightarrow$ True | False | P | Q | ...

Complex Sentence $\rightarrow$ (Sentence) | [Sentence]
| $\neg$Sentence
| Sentence $\wedge$ Sentence
...
| Sentence $\Leftrightarrow$ Sentence

We add operator precedence to disambiguate it

Operator precedence (from highest to lowest)
$\neg$, $\wedge$, $\vee$, $\implies$, $\Leftrightarrow$

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	true	*true*	*false*

This specific model: $m_1 = \{P_{1,2} = true, P_{2,2} = true, P_{3,1} = false\}$

(With these 3 symbols, $2^3 = 8$ possible models, feasible to enumerate.)

Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true **and**	S_2	is true
$S_1 \vee S_2$	is true iff	S_1	is true **or**	S_2	is true
$S_1 \implies S_2$	is true iff	S_1	is false **or**	S_2	is true
i.e.,	is false iff	S_1	is true **and**	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \implies S_2$	is true **and**	$S_2 \implies S_1$	is true

Simple recursive process evaluates an arbitrary sentence, e.g.,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

Truth Tables for Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Wumpus World Sentences in Propositional Logic

Let $P_{i,j}$ be true if there is a pit in $[i,j]$
Let $B_{i,j}$ be true if agent is in $[i,j]$ and perceives a breeze
Let $W_{i,j}$ be true if there is a wumpus in $[i,j]$
Let $S_{i,j}$ be true if agent is in $[i,j]$ and perceives a stench
... you can define other atomic sentences

Percept sentences part of KB:

No pit, no breeze in [1,1], but breeze perceived when in $[2,1]$

$$R_1 : \neg P_{1,1}$$
$$R_4 : \neg B_{1,1}$$
$$R_5 : B_{2,1}$$

Rules in KB:
"Pits cause breezes in adjacent squares" eqv. to "square is breezy **iff** adjacent pit"

Wumpus World Sentences in Propositional Logic

Let $P_{i,j}$ be true if there is a pit in $[i,j]$
Let $B_{i,j}$ be true if agent is in $[i,j]$ and perceives a breeze
Let $W_{i,j}$ be true if there is a wumpus in $[i,j]$
Let $S_{i,j}$ be true if agent is in $[i,j]$ and perceives a stench
... you can define other atomic sentences

Percept sentences part of KB:
No pit, no breeze in [1,1], but breeze perceived when in $[2,1]$

$$R_1 : \neg P_{1,1}$$
$$R_4 : \neg B_{1,1}$$
$$R_5 : B_{2,1}$$

Rules in KB:
"Pits cause breezes in adjacent squares" eqv. to "square is breezy **iff** adjacent pit"

$$R_2 : B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \vee P_{2,1})$$
$$R_3 : B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

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*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
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*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols); rows are possible models
if KB is true in a row/model, check that α is true; f not, entailment does not hold
If entailment not broken over all rows where KB is true, then else, α entailed

Depth-first enumeration of all models is sound and complete

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

    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])
```

```
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
         return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model))
and
                     TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

$O(2^n)$ for n symbols; problem is **co-NP-complete**

Proof Methods

Proof methods divide into (roughly) two kinds:

Model checking
truth table enumeration (always exponential in n)
improved backtracking, e.g., Davis–Putnam–Logemann–Loveland
backtracking with constraint propagation, backjumping
heuristic search in model space (sound but incomplete)
    e.g., min-conflicts-like hill-climbing algorithms

Theorem Proving/Deductive Systems: Application of inference rules
– Legitimate (sound) generation of new sentences from old
– Proof = a sequence of inference rule applications
    Can use inference rules as operators in a standard search alg.
– Typically require translation of sentences into a normal form

Two sentences are logically equivalent iff true in same models:
$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{array}{rcl}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha \quad \text{double-negation elimination} \\
(\alpha \implies \beta) & \equiv & (\neg\beta \implies \neg\alpha) \quad \text{contraposition} \\
(\alpha \implies \beta) & \equiv & (\neg\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \implies \beta) \land (\beta \implies \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) \quad \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

Validity and Satisfiability

A sentence is valid if it is true in **all** models,
e.g., $True$, $A \vee \neg A$, $A \implies A$, $(A \wedge (A \implies B)) \implies B$

Validity is connected to inference via the Deduction Theorem:
$KB \models \alpha$ if and only if $(KB \implies \alpha)$ is valid

A sentence is satisfiable if it is true in **some** model
e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in **no** models
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
i.e., prove α by *reductio ad absurdum*

Deductive Systems: Rules of Inference

◇ **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

◇ **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

◇ **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

◇ **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

◇ **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

◇ **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta}{\alpha}$$

◇ **Resolution**: (This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \qquad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Figure 6.13 Seven inference rules for propositional logic. The unit resolution rule is a special case of the resolution rule, which in turn is a special case of the full resolution rule for first-order logic discussed in Chapter 9.

Inference by Resolution

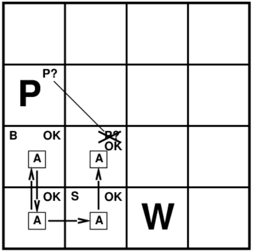
Conjunctive Normal Form (CNF—universal)
conjunction of **disjunctions** of **literals**

conjunction of **disjuncticlauses**

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

Resolution inference rule (for CNF): complete for propositional logic

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



where ℓ_i and m_j are complementary literals. 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

Conversion to CNF

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

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

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

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:

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

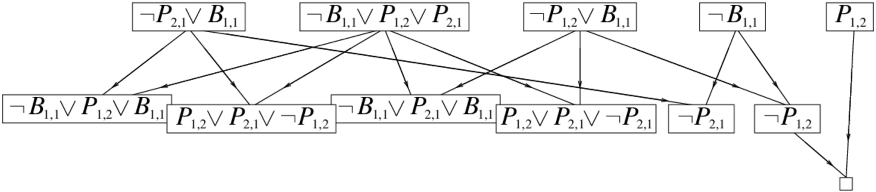
Proof by contradiction/refutation, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

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 $KB \wedge \neg\alpha$
  new ← { }
  loop do
    for each C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

Can actually use any search algorithm, with clauses as states and resolution as operators. Goal state is list of clauses containing empty clause.

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



Completeness of resolution algorithm follows from ground resolution theorem: If a set of clauses S is unsatisfiable, then the resolution closure $RC(S)$ of those clauses contains an empty clause.

$RC(S)$: set of all clauses derivable by repeated application of resolution rule to clauses in S or their derivatives.

Definite Clauses and Horn Clauses

Inference by resolution is complete, but sometimes an overkill

KB may contain restricted (rule-based) forms of sentences, such as:

Definite clause: disjunction of literals of which *exactly one* is positive.

$(\neg L_{1,1} \vee B_{1,1})$ is

$(P_{1,2} \vee P_{2,1})$ is not

$(\neg L_{1,1} \vee \neg B_{1,1})$ is not

Horn clause: disjunction of literals of which *at most one* is positive.

Which of the above is a Horn clause?

Negated literals $\neg A$ rewritten as $(A \implies \mathit{False})$ (integrity constraints)

Inference with Horn clauses can be done through forward chaining and backward chaining

These are more efficient than the resolution algorithm, run in linear time

Definite Clauses and Horn Clauses

Inference by resolution is complete, but sometimes an overkill

KB may contain restricted (rule-based) forms of sentences, such as:

Definite clause: disjunction of literals of which *exactly one* is positive.

$(\neg L_{1,1} \vee B_{1,1})$ is
$(P_{1,2} \vee P_{2,1})$ is not
$(\neg L_{1,1} \vee \neg B_{1,1})$ is not

Horn clause: disjunction of literals of which *at most one* is positive.

Which of the above is a Horn clause?

Negated literals $\neg A$ rewritten as $(A \implies \mathit{False})$ (integrity constraints)

Inference with Horn clauses can be done through forward chaining and backward chaining

These are more efficient than the resolution algorithm, run in linear time

Horn Form (restricted) KB (= **conjunction** of **Horn clauses**)
E.g., $C \wedge (B \implies A) \wedge (C \wedge D \implies B)$

Modus Ponens: complete for Horn KBs ($\alpha_1, \ldots, \alpha_n$ - premises, β - sought conclusion)

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \implies \beta}{\beta}$$

Known as forward chaining inference rule; repeated applications until sentence of interest obtained – forward chaining algorithm

Modus Tollens - a form of Modus Ponens

$$\frac{\neg\beta, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \implies \beta}{\neg(\alpha_1 \wedge \cdots \wedge \alpha_n)}$$

Known as backward chaining inference rule; repeated applications until all premises obtained – backward chaining algorithm

Both algorithms intuitive and run in **linear** time

Inference via forward or backward chaining forms basis of logic programming (Chapter 9)

Idea: Add literals in KB to facts (satisfied premises)
apply each premise satisfied in KB (fire rules)
add rule's conclusion as new fact/premise to the KB
(this is inference propagation via forward chaining)
stop when query found as fact or no more inferences

$P \implies Q$

$L \land M \implies P$

$B \land L \implies M$

$A \land P \implies L$

$A \land B \implies L$

A

B

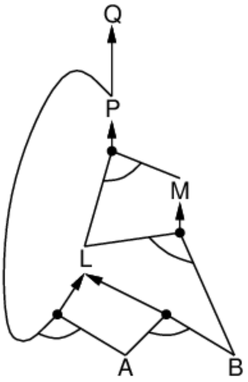
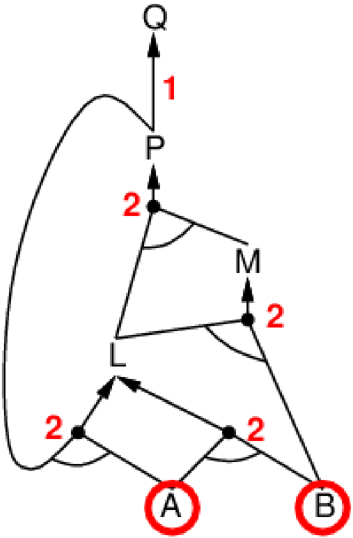


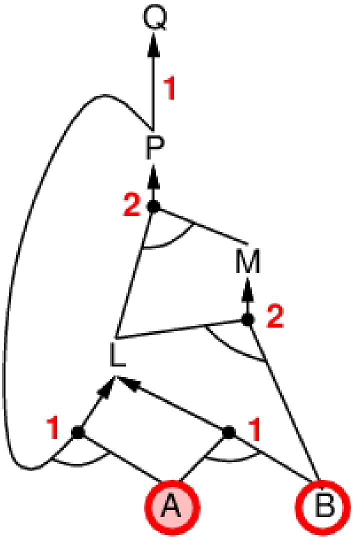
Figure: AND-OR tree

Forward Chaining Algorithm

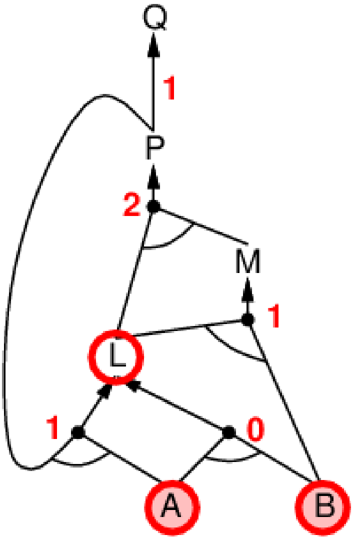
function PL-FC-ENTAILS?(*KB*, *q*) **returns** *true* or *false*
  inputs: *KB*, the knowledge base, a set of propositional Horn clauses
    q, the query, a proposition symbol
  local variables: *count*, table indexed by clause, initial nr. of premises
    inferred, table indexed by symbol, entries initially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

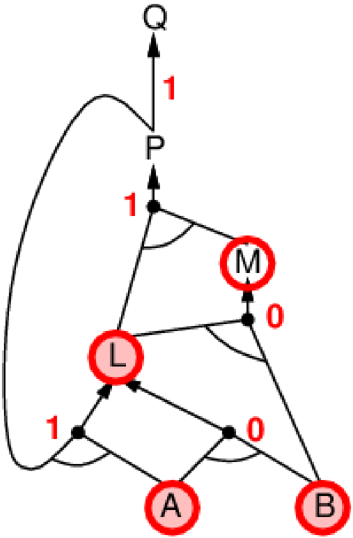
  while *agenda* is not empty **do**
    $p \leftarrow$ POP(*agenda*)
    unless *inferred*[p] **do**
      inferred[p] $\leftarrow$ *true*
      for each Horn clause c in whose premise p appears **do**
        decrement *count*[c]
        if *count*[c] $= 0$ **then do**
          if HEAD[c] $= q$ **then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


Q
1
P
2
M
2
L
2
2
A
B

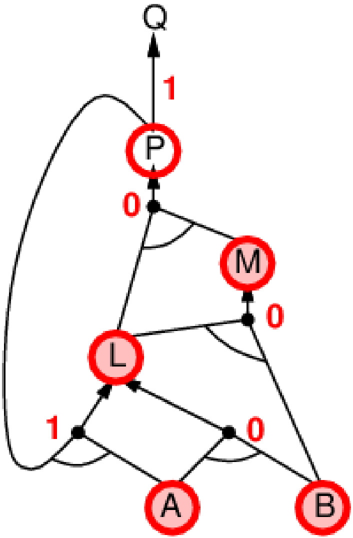

Q
1
P
2
M
2
L
1
1
A
B

Forward Chaining Example

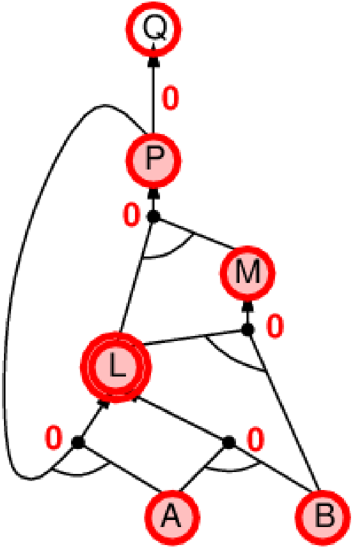


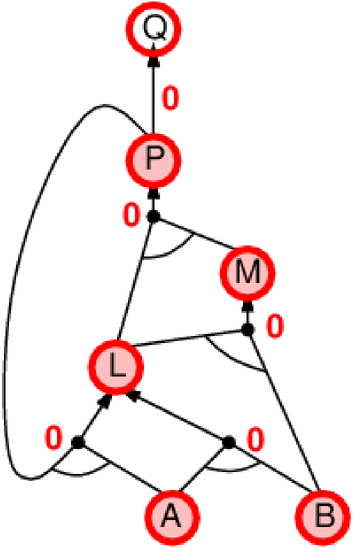

Q
1
P
1
M
0
L
1
0
A
B

Forward Chaining Example

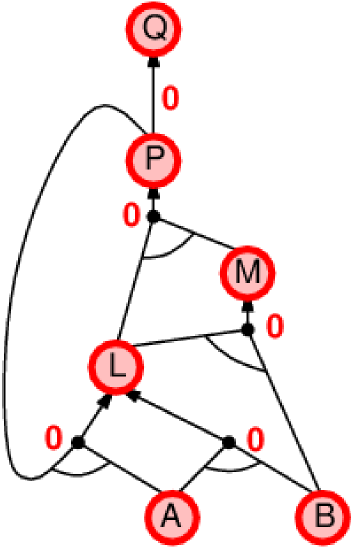


Forward Chaining Example




Q
0
P
0
M
0
L
0
0
A
B

Forward Chaining Example



Proof of Completeness

FC derives every atomic sentence that is entailed by KB

1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model m, assigning true/false to symbols
3. Every clause in the original KB is true in m

Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \;\Rightarrow b$ is false in m

Then $a_1 \wedge \ldots \wedge a_k$ is true in m and b is false in m

Therefore the algorithm has not reached a fixed point!

4. Hence m is a model of KB
5. If $KB \models q$, q is true in **every** model of KB, including m

General idea: construct any model of KB by sound inference, check α

FC is an example of a data-driven reasoning algorithm

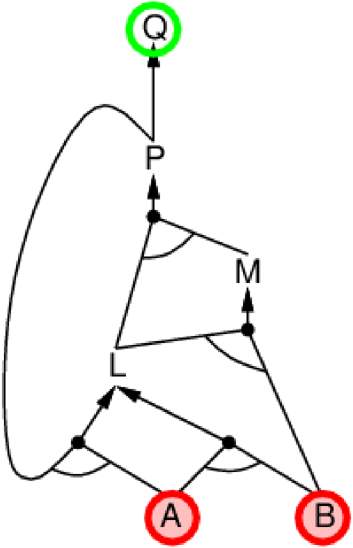
start with what known, derive new conclusions, with no particular goal in mind

Idea: goal-driven reasoning – work backwards from the query q:
to prove q by BC,
check if q is known already, or
prove by BC all premises of some rule concluding q

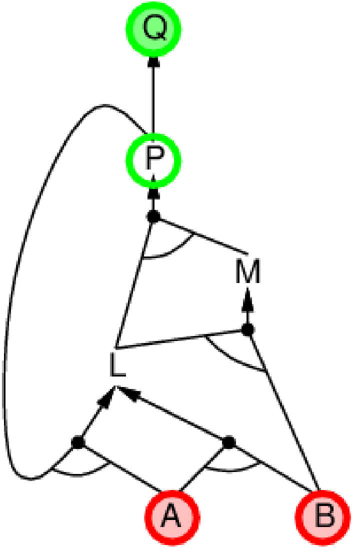
Avoid loops: check if new subgoal is already on the goal stack

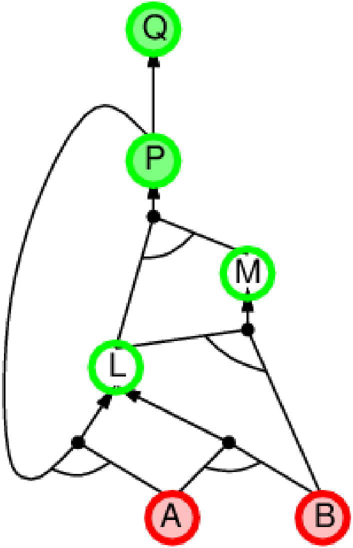
Avoid repeated work: check if new subgoal
1) has already been proved true, or
2) has already failed

Backward Chaining Example

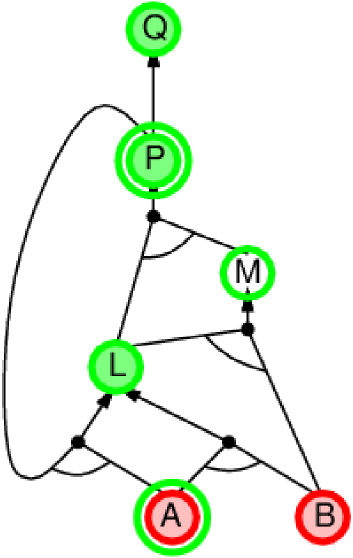


Backward Chaining Example

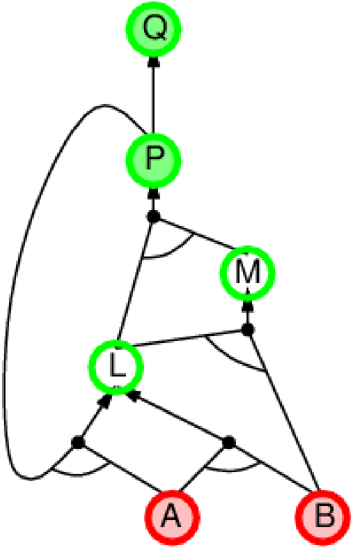


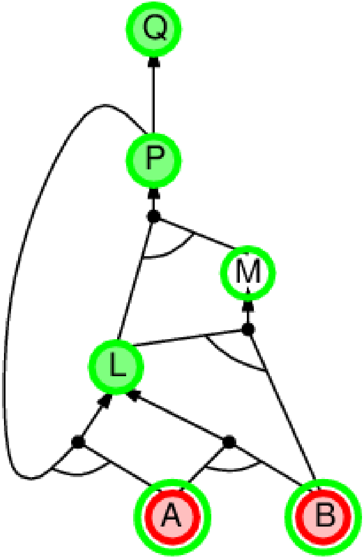

Q
P
M
L
A
B

Backward Chaining Example

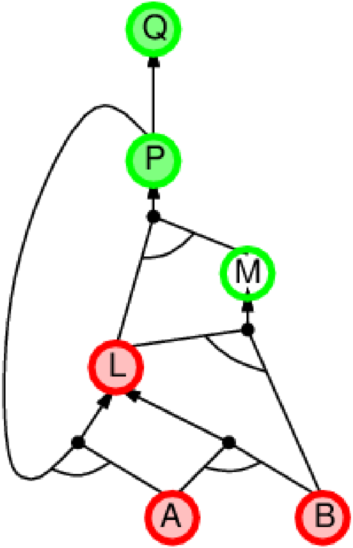


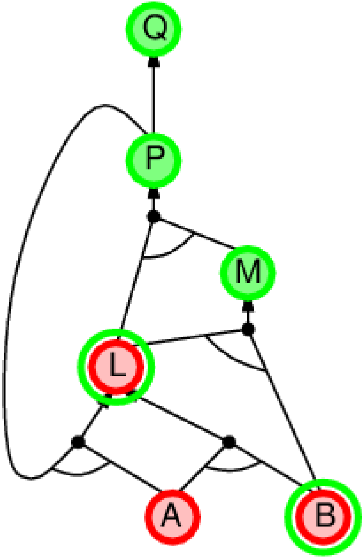
Backward Chaining Example



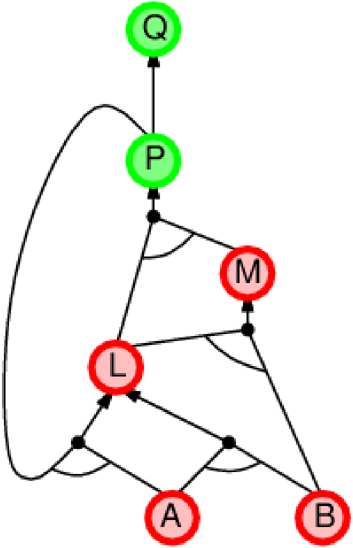

Q
P
M
L
A
B

Backward Chaining Example

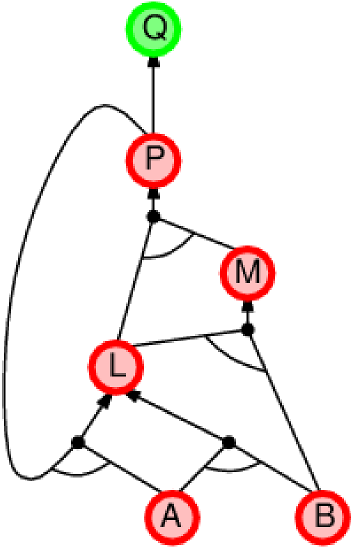



Q
P
M
L
A
B

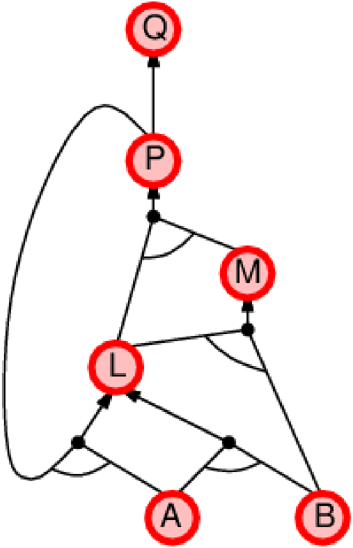
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



Forward versus Backward Chaining

FC is data-driven, 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, 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, because only relevant facts are touched

A wumpus-world agent using propositional logic:

$$\neg P_{1,1}$$
$$\neg W_{1,1}$$
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$
$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$$
$$\neg W_{1,1} \vee \neg W_{1,2}$$
$$\neg W_{1,1} \vee \neg W_{1,3}$$
$$\ldots$$

64 distinct proposition symbols, 155 sentences

PL-WUMPUS-AGENT Algorithm

function PL-WUMPUS-AGENT(*percept*) **returns** an *action*
  inputs: *percept*, a list, [*stench*,*breeze*,*glitter*]
  static: *KB*, initially containing the "physics" of the wumpus world
    x, y, *orientation*, the agent's position (init. [1,1]) and orient. (init. *right*)
    visited, an array indicating which squares have been visited, initially *false*
    action, the agent's most recent action, initially null
    plan, an action sequence, initially empty

  update x,y,*orientation*, *visited* based on *action*
  if *stench* **then** TELL(KB, $S_{x,y}$) **else** TELL(KB, $\neg\ S_{x,y}$)
  if *breeze* **then** TELL(KB, $B_{x,y}$) **else** TELL(KB, $\neg\ B_{x,y}$)
  if *glitter* **then** *action* ← *grab*
  else if *plan* is nonempty then *action* ← POP(*plan*)
  else if for some fringe square [i,j], ASK(KB, $(\neg\ P_{i,j} \wedge\ \neg\ W_{i,j})$) is *true* **or**
    for some fringe square [i,j], ASK(KB, $(P_{i,j} \vee\ W_{i,j})$) is *false* **then do**
   plan ← A*-GRAPH-SEARCH(ROUTE-PB([x,y], *orientation*, [i,j], *visited*))
   action ← POP(*plan*)
  else *action* ← a randomly chosen move
  return *action*

Propositional Logic 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: truth of one sentence given another
– inference: deriving sentences from other sentences
– soundess: 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 does not scale to environments of unbounded size, as it lacks expressive power to deal concisely with time, space, and universal patterns of relationships among objects