Artificial Intelligence- Logical Agents

Amir Aavani Amir@Aavani.net
http://Amir.Aavani.net/CS/AI-86

7. Dezember 2007

Outline

Outline

Outline

1. Knowledge-based Agents
2. Wampus world
3. Logic in general
4. Propositioanl Logic

Knowledge-Based Agents

1. Knowledge Base(KB): Set of sentences in knowledge representation language.
2. Each sentence represent some assertion about the world.

Knowledge-Based Agents

1. Knowledge Base(KB): Set of sentences in knowledge representation language.
2. Each sentence represent some assertion about the world.
3. Tell method is a way to add new statements to KB.

Knowledge-Based Agents

1. Knowledge Base(KB): Set of sentences in knowledge representation language.
2. Each sentence represent some assertion about the world.
3. Tell method is a way to add new statements to KB.
4. Ask method is a way to query from KB.

Knowledge-Based Agents

1. Knowledge Base(KB): Set of sentences in knowledge representation language.
2. Each sentence represent some assertion about the world.
3. Tell method is a way to add new statements to KB.
4. Ask method is a way to query from KB.
5. Inference: deriving new sentences from old, using logic.

Knowledge-Based Agents, Cont

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

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

Knowledge-Based Agents, Cont

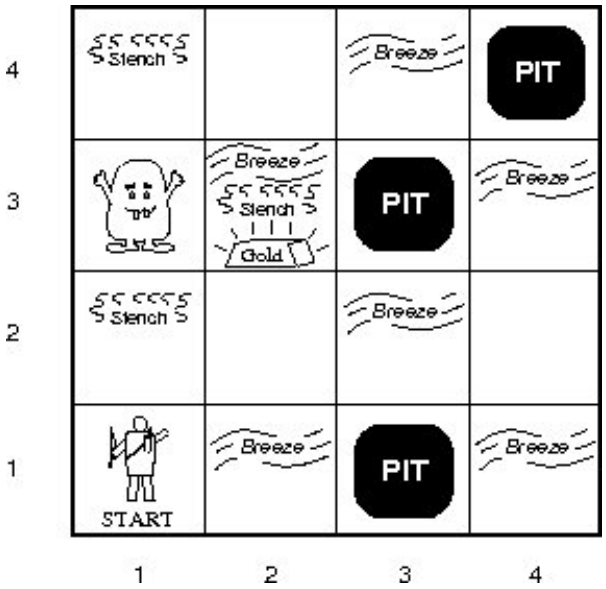
1. In Knowledge level:, one needs to specify what the agent knows and what its goals are.

Knowledge-Based Agents, Cont

1. In Knowledge level:, one needs to specify what the agent knows and what its goals are.
2. In Implementation Level:, one talks about data structures and algorithms used to work with statements.

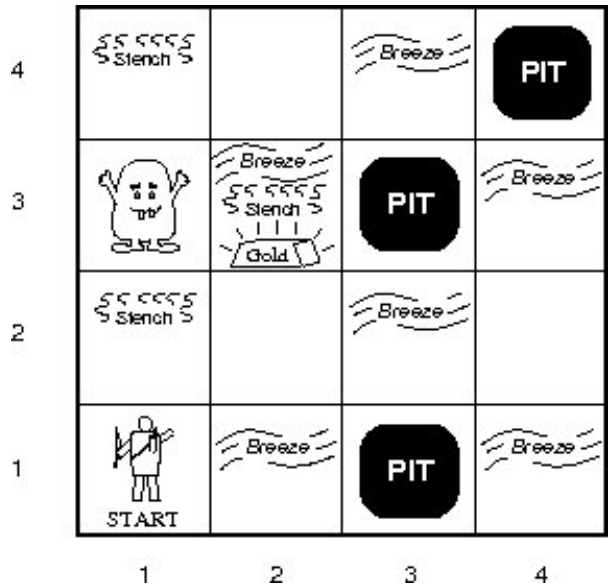
Wampus world, PEAS

1. Performance measure:
   . gold: +1000, death: -1000
   . -1 per step, -10 for using arrow.
2. Environment:
   . A 4x4 grid of rooms.
   . Agent starts in squeare [1,1].
   . There may be a pit in each square.
   . etc.
3. Actuators: Turn left, Turn right, Forward
   . Forward, Grab, Release, Shoot
4. Sensors: Stench, Breeze, Glitter, Bump, Scream



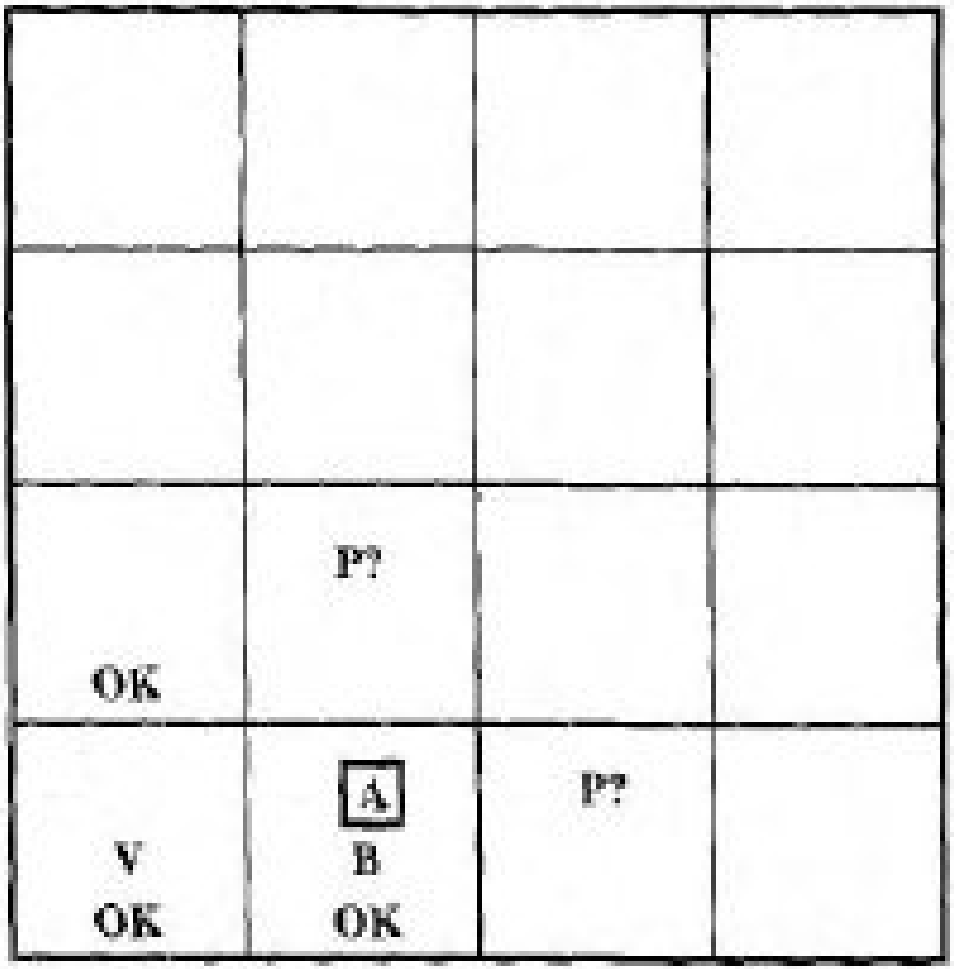
Wampus world, exploration

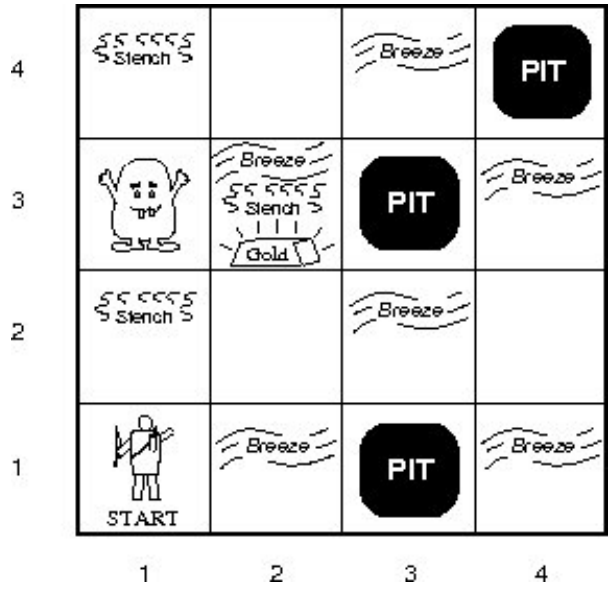
OK			
OK A	OK		



A: [-,-,-,-,-]

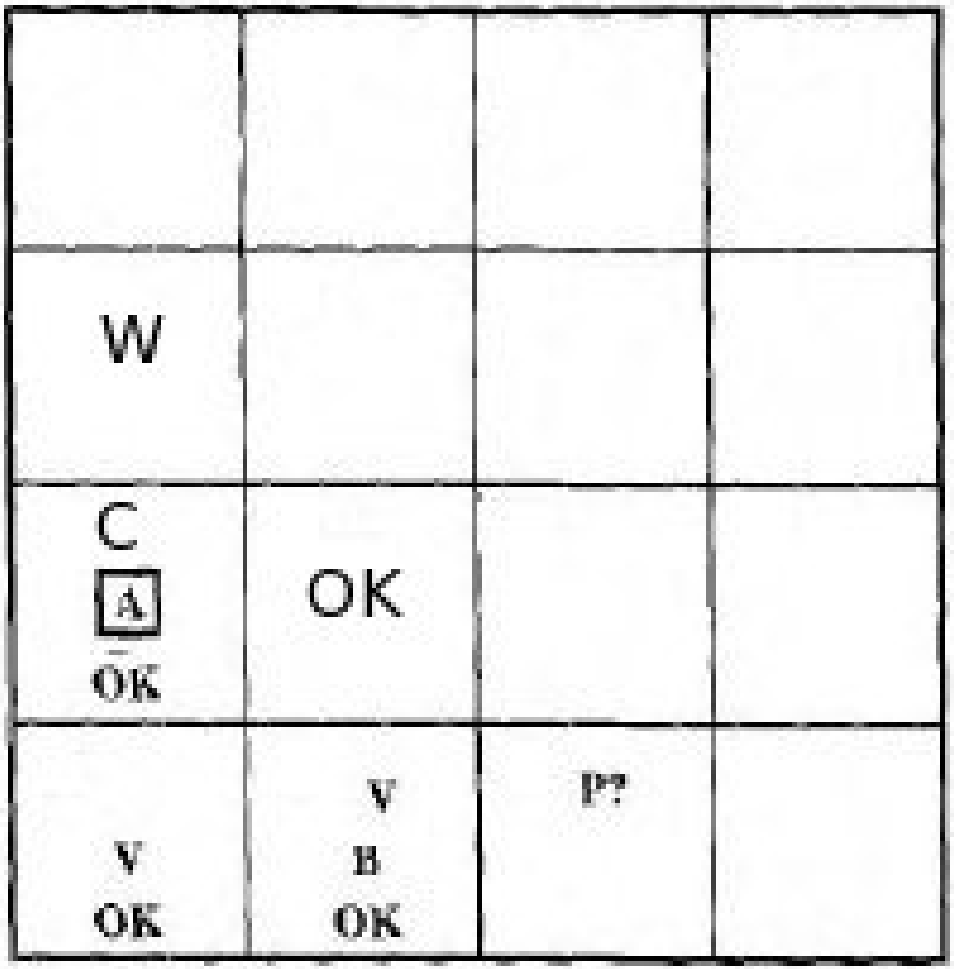
Wampus world, exploration

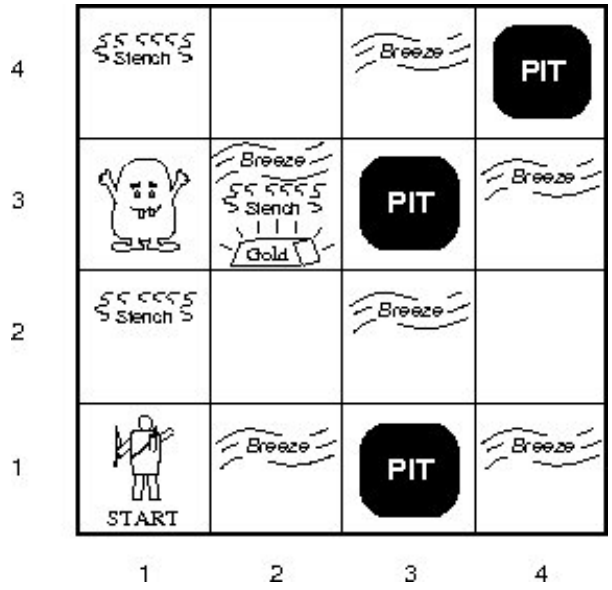




B: [-,B,-,-,-]

Wampus world, exploration





C: [S,-,-,-,-]

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   . $\Rightarrow$No safe action

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   . ⇒No safe action
   . Assuming pits uniformly distributed,
   . (2,2) has pit with prob. 0.86.
2. Consider if we have Smell in (1,1)

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   . ⇒No safe action
   . Assuming pits uniformly distributed,
   . (2,2) has pit with prob. 0.86.
2. Consider if we have Smell in (1,1)
   . ⇒No neighbor is safe

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   - $\Rightarrow$No safe action
   - Assuming pits uniformly distributed,
   - (2,2) has pit with prob. 0.86.
2. Consider if we have Smell in (1,1)
   - $\Rightarrow$No neighbor is safe
   - Shoot

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   - ⇒No safe action
   - Assuming pits uniformly distributed,
   - (2,2) has pit with prob. 0.86.
2. Consider if we have Smell in (1,1)
   - ⇒No neighbor is safe
   - Shoot
   - Wampus was there ⇒ dead ⇒ safe

Wampus world, exploration

1. Consider if we have Breeze in both (2,1) and (1,2)
   - ⇒No safe action
   - Assuming pits uniformly distributed,
   - (2,2) has pit with prob. 0.86.
2. Consider if we have Smell in (1,1)
   - ⇒No neighbor is safe
   - Shoot
   - Wampus was there ⇒ dead ⇒ safe
   - Wampus wasn't there ⇒ safe

Logic in general

1. Logics are formal languages for representing information.
2. To define a logic, its syntax and semantics should be defined.

Logic in general

1. Logics are formal languages for representing information.
2. To define a logic, its syntax and semantics should be defined.
3. Syntax defines the structure of sentences.

Logic in general

1. Logics are formal languages for representing information.
2. To define a logic, its syntax and semantics should be defined.
3. Syntax defines the structure of sentences.
4. Semantic defines the meaning of sentences.

Entailment

1. A Entails B (informally) means that B follows logically from B.
2. $KB \models \alpha$ (Knowledge base KB entails α)
   . if and only if
   α is true in all worlds(models) where KB is true.

Entailment

1. A Entails B (informally) means that B follows logically from B.
2. $KB \models \alpha$ (Knowledge base KB entails α)
   . if and only if
   α is true in all worlds(models) where KB is true.
3. $KB = \{\text{It is cold, It is raining}\} \models \text{It is cold and rainy.}$

Models

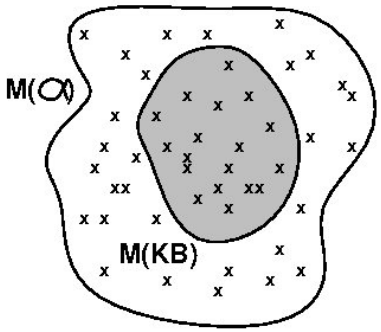
1. m is a model for a sentence α if α is true in m.

Models

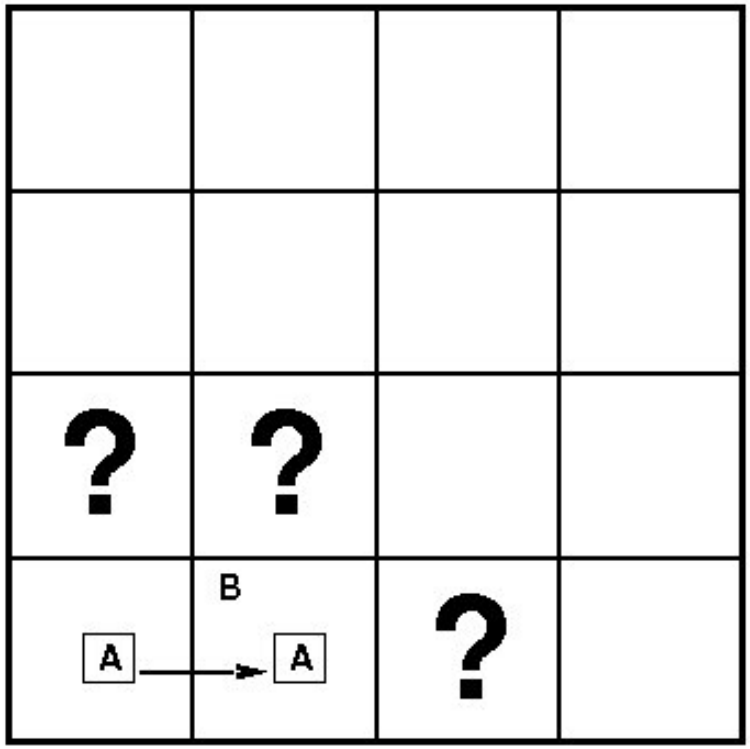
1. m is a model for a sentence α if α is true in m.
2. $M(\alpha)$ is the set of all models of α.

Models

1. m is a model for a sentence α if α is true in m.
2. $M(\alpha)$ is the set of all models of α.
3. $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

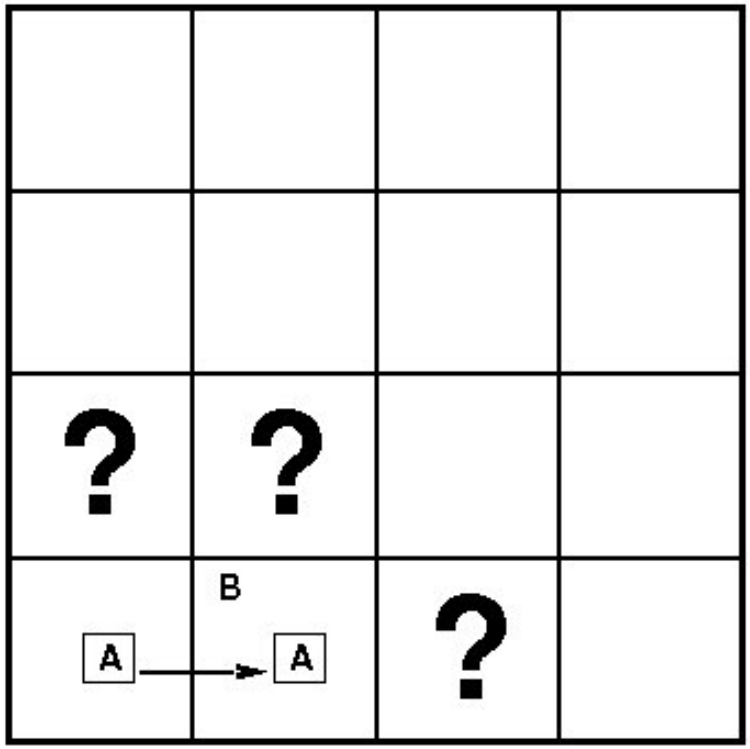


Wampus models



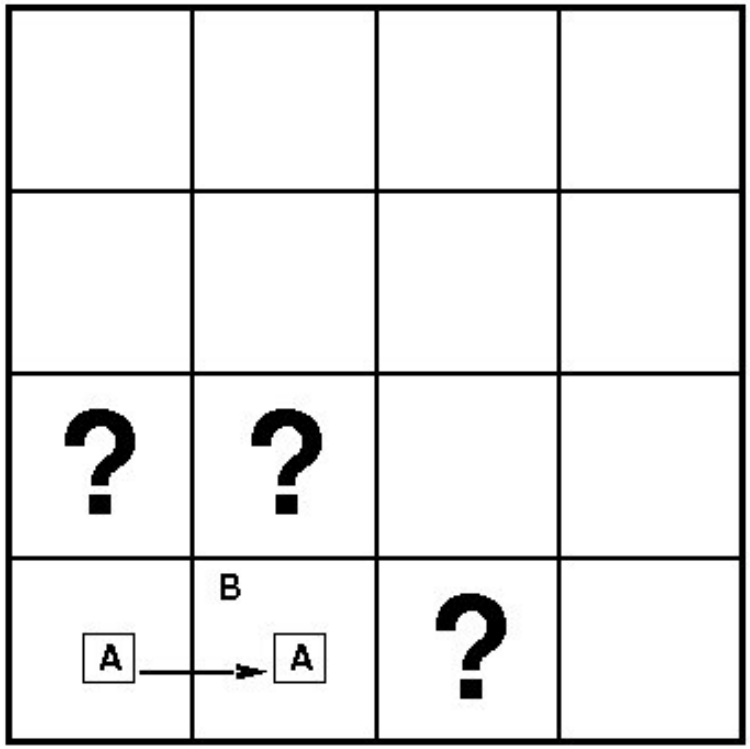
1. Nothing detected in [1,1], and breeze in [2,1].

Wampus models



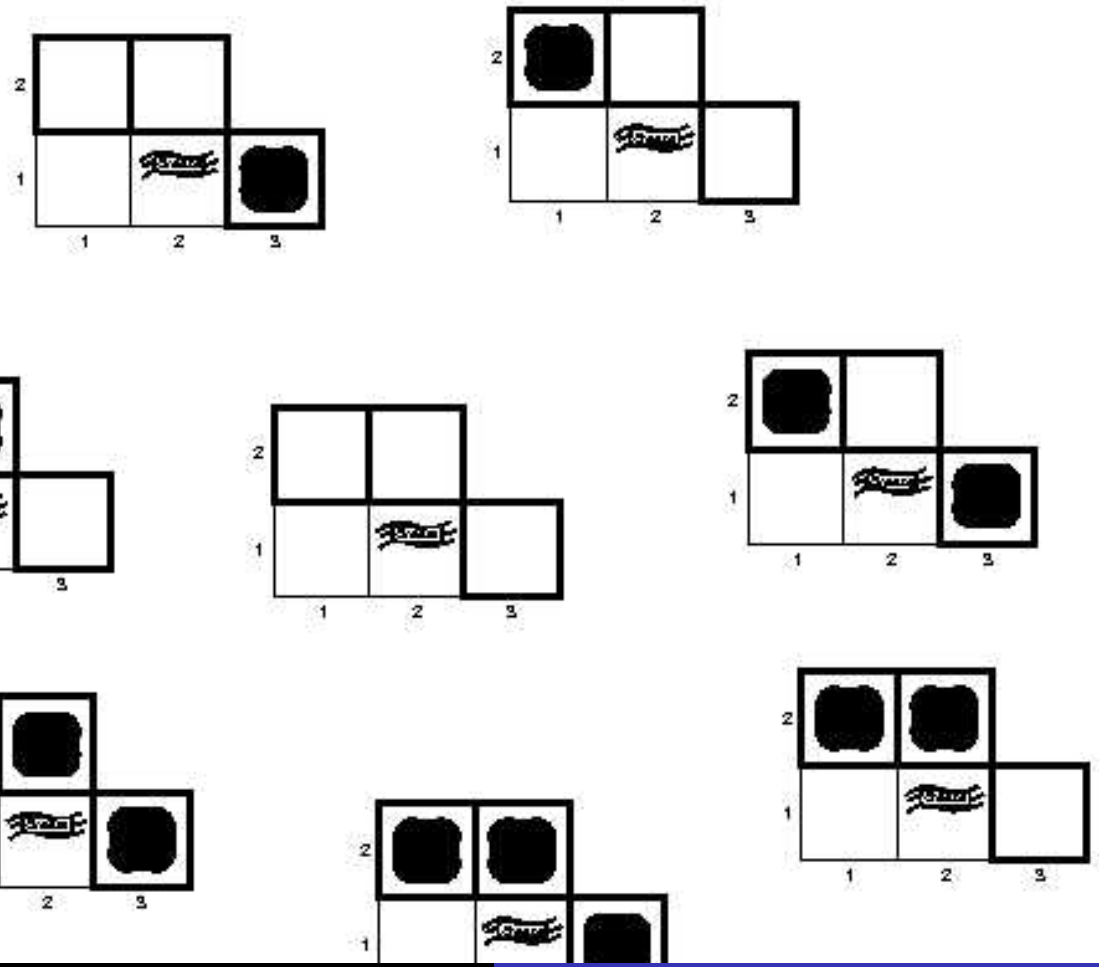
1. Nothing detected in [1,1], and breeze in [2,1].
2. We want to find out if ? are pit or not.

Wampus models

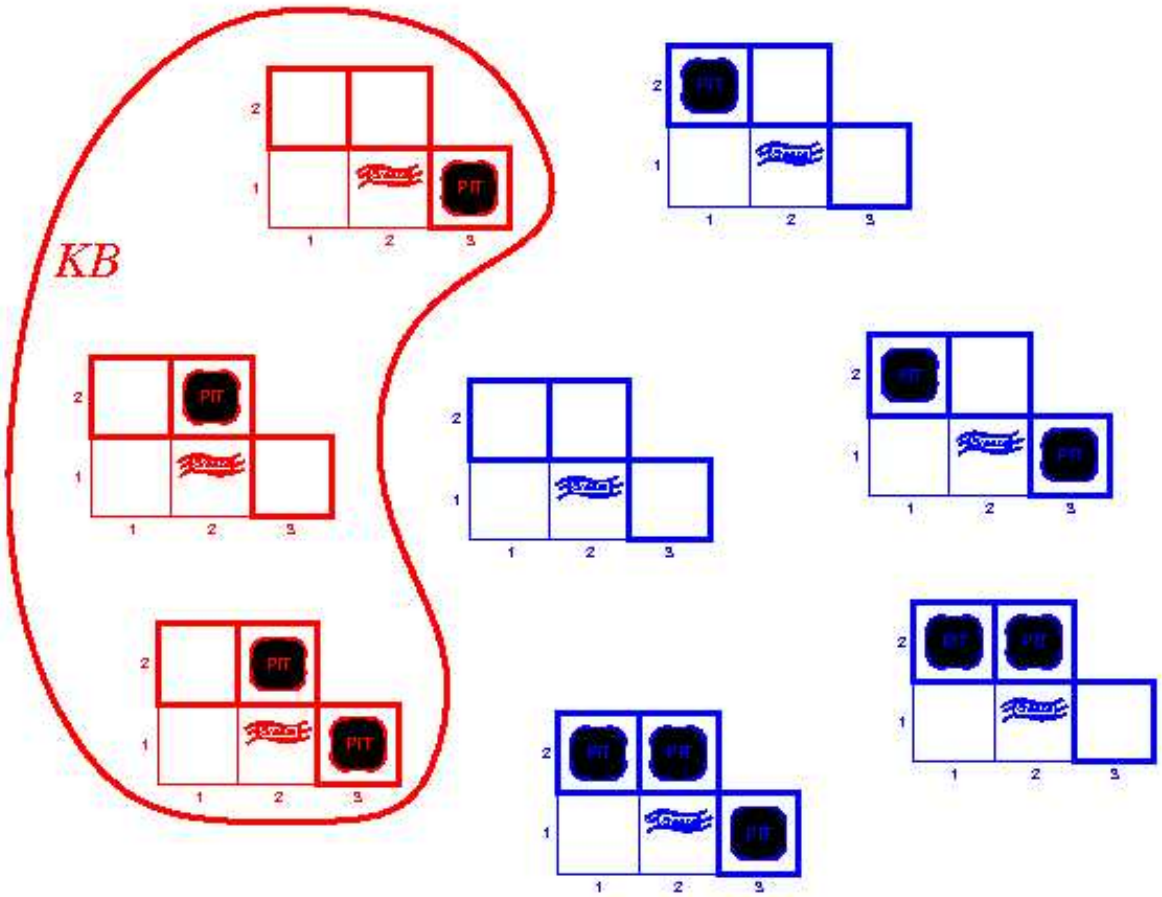


1. Nothing detected in [1,1], and breeze in [2,1].
2. We want to find out if ? are pit or not.
3. There are 8 modals.

Wampus models, Cont

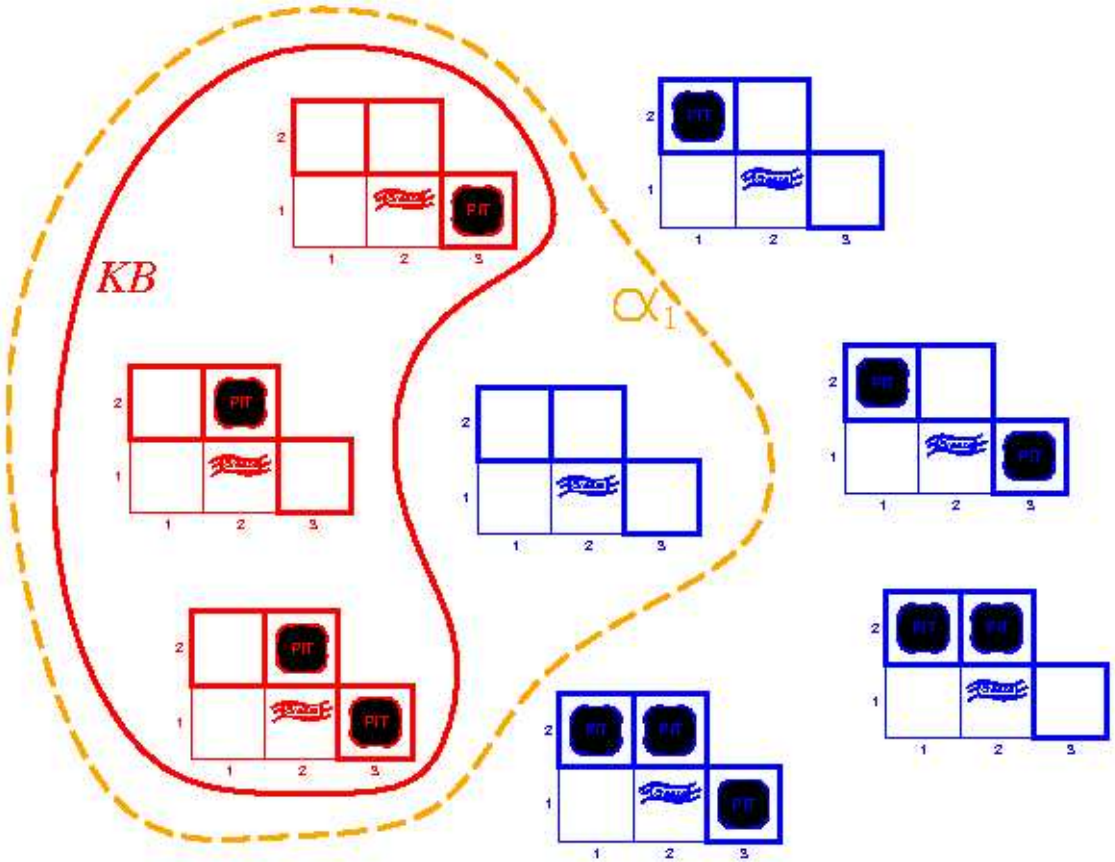


Wampus models, Cont



KB= wumpus-world rules + observations

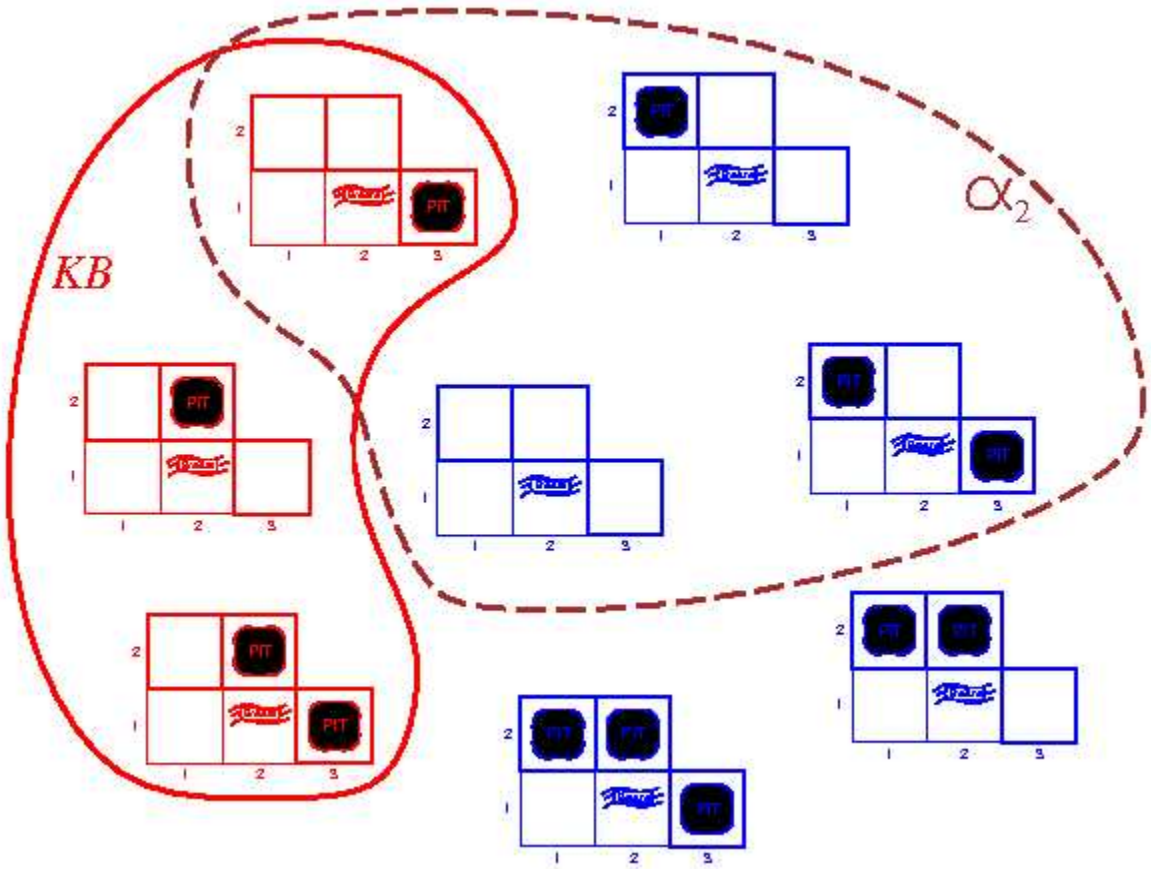
Wampus models, Cont



KB= wumpus-world rules + observations

α_1 = [1,2] is safe

Wampus models, Cont



KB= wumpus-world rules + observations

α_2 = [2,2] is safe.

Inference

1. Soundness= Truth preserving.

Inference

1. Soundness= Truth preserving.
2. Completeness= A logic is complete if it can derive any sentence that is entailed.

Inference

1. Soundness= Truth preserving.
2. Completeness= A logic is complete if it can derive any sentence that is entailed.
3. A Complete and Sound inference procedure will answer any question whose answer follows from what is known by the KB.

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol
3. Symbol $\rightarrow$ P|Q|R|$\cdots$

Propositional logic- Syntax

1. Sentence $\longrightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\longrightarrow$ True | False | Symbol
3. Symbol $\longrightarrow$ P|Q|R|$\cdots$
4. ComplexSentence $\longrightarrow \neg$ Sentence

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol
3. Symbol $\rightarrow$ P|Q|R|$\cdots$
4. ComplexSentence $\rightarrow \neg$ Sentence
5. ComplexSentence $\rightarrow$ (Sentence $\wedge$ Sentence)

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol
3. Symbol $\rightarrow$ P|Q|R|$\cdots$
4. ComplexSentence $\rightarrow \neg$ Sentence
5. ComplexSentence $\rightarrow$ (Sentence $\wedge$ Sentence)
6. ComplexSentence $\rightarrow$ (Sentence $\vee$ Sentence)

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol
3. Symbol $\rightarrow$ P|Q|R|$\cdots$
4. ComplexSentence $\rightarrow \neg$ Sentence
5. ComplexSentence $\rightarrow$ (Sentence $\wedge$ Sentence)
6. ComplexSentence $\rightarrow$ (Sentence $\vee$ Sentence)
7. ComplexSentence $\rightarrow$ (Sentence $\Rightarrow$ Sentence)

Propositional logic- Syntax

1. Sentence $\rightarrow$ Atomic Sentence | ComplexSentence
2. AtomicSentence $\rightarrow$ True | False | Symbol
3. Symbol $\rightarrow$ P|Q|R|$\cdots$
4. ComplexSentence $\rightarrow \neg$ Sentence
5. ComplexSentence $\rightarrow$ (Sentence $\wedge$ Sentence)
6. ComplexSentence $\rightarrow$ (Sentence $\vee$ Sentence)
7. ComplexSentence $\rightarrow$ (Sentence $\Rightarrow$ Sentence)
8. ComplexSentence $\rightarrow$ (Sentence $\Leftrightarrow$ Sentence)

Propositional logic- Semantics

1. Every statement in a modal is either True or False.

Propositional logic- Semantics

1. Every statement in a modal is either True or False.
2. $\neg S$ is true iff S is false.

Propositional logic- Semantics

1. Every statement in a modal is either True or False.
2. $\neg S$ is true iff S is false.
3. $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true.

Propositional logic- Semantics

1. Every statement in a modal is either True or False.
2. $\neg S$ is true iff S is false.
3. $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true.
4. $S_1 \vee S_2$ is true iff S_1 is true or S_2 is true.

Propositional logic- Semantics

1. Every statement in a modal is either True or False.
2. $\neg S$ is true iff S is false.
3. $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true.
4. $S_1 \vee S_2$ is true iff S_1 is true or S_2 is true.
5. $S_1 \Rightarrow S_2$ is true iff S_1 is true and S_2 is false.

Propositional logic- Semantics

1. Every statement in a modal is either True or False.
2. $\neg S$ is true iff S is false.
3. $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true.
4. $S_1 \vee S_2$ is true iff S_1 is true or S_2 is true.
5. $S_1 \Rightarrow S_2$ is true iff S_1 is true and S_2 is false.
6. $S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_1 \Rightarrow S_2$ is true.

Wampus world with propositional logic

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

Wampus world with propositional logic

1. $P_{i,j}$ is true if there is a pit in [i,j].
2. $B_{i,j}$ is true if there is a breeze in [i,j].

Wampus world with propositional logic

1. $P_{i,j}$ is true if there is a pit in [i,j].
2. $B_{i,j}$ is true if there is a breeze in [i,j].
3. Pits cause breezes in adjanced squares. i.e.

Wampus world with propositional logic

1. $P_{i,j}$ is true if there is a pit in [i,j].
2. $B_{i,j}$ is true if there is a breeze in [i,j].
3. Pits cause breezes in adjanced squares. i.e.
4. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
5. $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Wampus world with propositional logic, Cont

1. $R_1 : \neg P_{1,1}$

Wampus world with propositional logic, Cont

1. $R_1 : \neg P_{1,1}$
2. $R_2 : \neg B_{1,1}$

Wampus world with propositional logic, Cont

1. $R_1 : \neg P_{1,1}$
2. $R_2 : \neg B_{1,1}$
3. $R_3 : B_{2,1}$

Wampus world with propositional logic, Cont

1. $R_1 : \neg P_{1,1}$
2. $R_2 : \neg B_{1,1}$
3. $R_3 : B_{2,1}$
4. $R_4 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

Wampus world with propositional logic, Cont

1. $R_1 : \neg P_{1,1}$
2. $R_2 : \neg B_{1,1}$
3. $R_3 : B_{2,1}$
4. $R_4 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
5. $R_5 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Inference by enumeration

```
function TT-ENTAILS? (KB, a) returns true or false
 symbols= list of the symbols in KB and a.
 return TT-CHECK-ALL (KB, a, symbols, []);

 function TT-CHECK-ALL (KB, a, symbols, models)
                        returns true or false
   if Empty(symbols) then
     if PL-TRUE?(KB,models) then return PL-TRUE?(a,models)
     else return false;
   else
     P= FIRST(symbols); rest= REST(symbols);
     return TT-CHECK-ALL(KB,a,rest,EXTEND(P,true,modal))
      and TT-CHECK-ALL(KB,a,rest,EXTEND(P,false,modal))
```

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
2. $(a \wedge b) \equiv (b \wedge a)$ commutativity of $\wedge$
3. $(a \vee b) \equiv (b \vee a)$ commutativity of $\vee$

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
2. $(a \wedge b) \equiv (b \wedge a)$ commutativity of $\wedge$
3. $(a \vee b) \equiv (b \vee a)$ commutativity of $\vee$
4. $((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$ associativity of $\wedge$
5. $((a \vee b) \vee c) \equiv (a \vee (b \vee c))$ associativity of $\vee$

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
2. $(a \wedge b) \equiv (b \wedge a)$ commutativity of $\wedge$
3. $(a \vee b) \equiv (b \vee a)$ commutativity of $\vee$
4. $((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$ associativity of $\wedge$
5. $((a \vee b) \vee c) \equiv (a \vee (b \vee c))$ associativity of $\vee$
6. $(\neg(\neg a)) = a$ double-negation elimination

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
2. $(a \wedge b) \equiv (b \wedge a)$ commutativity of $\wedge$
3. $(a \vee b) \equiv (b \vee a)$ commutativity of $\vee$
4. $((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$ associativity of $\wedge$
5. $((a \vee b) \vee c) \equiv (a \vee (b \vee c))$ associativity of $\vee$
6. $(\neg(\neg a)) = a$ double-negation elimination
7. $(\neg(a \wedge b)) \equiv (\neg a \vee \neg b)$ De Morgan
8. $(\neg(a \vee b)) \equiv (\neg a \wedge \neg b)$ De Morgan

Some concepts in propositional logic

1. Logical Equivalancy: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
2. $(a \wedge b) \equiv (b \wedge a)$ commutativity of $\wedge$
3. $(a \vee b) \equiv (b \vee a)$ commutativity of $\vee$
4. $((a \wedge b) \wedge c) \equiv (a \wedge (b \wedge c))$ associativity of $\wedge$
5. $((a \vee b) \vee c) \equiv (a \vee (b \vee c))$ associativity of $\vee$
6. $(\neg(\neg a)) = a$ double-negation elimination
7. $(\neg(a \wedge b)) \equiv (\neg a \vee \neg b)$ De Morgan
8. $(\neg(a \vee b)) \equiv (\neg a \wedge \neg b)$ De Morgan
9. $(a \wedge (b \vee c)) \equiv (a \wedge b) \vee (a \wedge c)$ distributivity of $\wedge$ over $\vee$
10. $(a \vee (b \wedge c)) \equiv (a \vee b) \wedge (a \vee c)$ distributivity of $\vee$ over $\wedge$

Some concepts in propositional logic, Cont

1. Validity: P is valid iff it it is true in all models.

Some concepts in propositional logic, Cont

1. Validity: P is valid iff it it is true in all models.
2. Satisfiablity: P is satisfiable if it is true in some models.

Some concepts in propositional logic, Cont

1. Validity: P is valid iff it it is true in all models.
2. Satisfiablity: P is satisfiable if it is true in some models.
3. Unsatisfiablity: P is unsatisfiable if it is true in no models.

Some concepts in propositional logic, Cont

1. Validity: P is valid iff it it is true in all models.
2. Satisfiablity: P is satisfiable if it is true in some models.
3. Unsatisfiablity: P is unsatisfiable if it is true in no models.
4. $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is unsatifiable.

Reasoning patterns in propositional logic

1. One can use equivalancy to convert one formula to another.

Reasoning patterns in propositional logic

1. One can use equivalancy to convert one formula to another.
2. Equivalancy is not strong enough to be able to prove (some) new statements.

Reasoning patterns in propositional logic

1. One can use equivalancy to convert one formula to another.
2. Equivalancy is not strong enough to be able to prove (some) new statements.
3. Inference rules can generate new statements.

Reasoning patterns in propositional logic

1. One can use equivalancy to convert one formula to another.
2. Equivalancy is not strong enough to be able to prove (some) new statements.
3. Inference rules can generate new statements.
4. MP: $\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$

Reasoning patterns in propositional logic

1. One can use equivalancy to convert one formula to another.
2. Equivalancy is not strong enough to be able to prove (some) new statements.
3. Inference rules can generate new statements.
4. MP: $\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
5. And-Elimination: $\frac{\alpha \wedge \beta}{\alpha}$

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.
3. $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_6.

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.
3. $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_6.
4. $R_8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})))$ from R_7.

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.
3. $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_6.
4. $R_8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})))$ from R_7.
5. $R_9 : \neg(P_{1,2} \vee P_{2,1})))$ from R_8, R_2 by MP.

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.
3. $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_6.
4. $R_8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})))$ from R_7.
5. $R_9 : \neg(P_{1,2} \vee P_{2,1})))$ from R_8, R_2 by MP.
6. $R_{10} : (\neg P_{1,2} \wedge \neg P_{2,1})$ from R_9.

Reasoning patterns in propositional logic, Cont

1. $P_{1,2}$??
2. $R_6 : ((B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_4.
3. $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}))$ from R_6.
4. $R_8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})))$ from R_7.
5. $R_9 : \neg(P_{1,2} \vee P_{2,1})))$ from R_8, R_2 by MP.
6. $R_{10} : (\neg P_{1,2} \wedge \neg P_{2,1})$ from R_9.
7. $R_{11} : \neg P_{1,2}$ from R_{10}.

Reasoning patterns in propositional logic, Cont

1. Proof: a sequence of application of inference rules.

Reasoning patterns in propositional logic, Cont

1. Proof: a sequence of application of inference rules.
2. Finding a proof is a search problem.

Reasoning patterns in propositional logic, Cont

1. Proof: a sequence of application of inference rules.
2. Finding a proof is a search problem.
3. Problem Statement: ?
4. Goal Statement: ?
5. Successor Function: ?
6. Path Cost: ?

Reasoning patterns in propositional logic, Cont

1. A Horn clause: is disjunction of literals of which atmost one is positive.
2. $\neg l_1 \vee \neg l_2 \cdots \neg l_k \vee l_{k+1} \equiv l_1 \wedge l_2 \cdots \wedge l_k \Rightarrow l_{k+1}$

Reasoning patterns in propositional logic, Cont

1. A Horn clause: is disjunction of literals of which atmost one is positive.
2. $\neg l_1 \vee \neg l_2 \cdots \neg l_k \vee l_{k+1} \equiv l_1 \wedge l_2 \cdots \wedge l_k \Rightarrow l_{k+1}$
3. There are two algorithms to work with Horn clauses(forward checking and backward checking).

Reasoning patterns in propositional logic, Cont

1. A Horn clause: is disjunction of literals of which atmost one is positive.
2. $\neg l_1 \vee \neg l_2 \cdots \neg l_k \vee l_{k+1} \equiv l_1 \wedge l_2 \cdots \wedge l_k \Rightarrow l_{k+1}$
3. There are two algorithms to work with Horn clauses(forward checking and backward checking).
4. Deciding entailment with Horn clauses can be done in linear time.

Forward chaining

1. Query in FC is a symbol.

Forward chaining

1. Query in FC is a symbol.
2. Generate all sybmols that can be inferenced (add them to KB) till the query found.

Forward chaining

1. Query in FC is a symbol.
2. Generate all sybmols that can be inferenced (add them to KB) till the query found.
3. Start with known facts (positive literals).

Forward chaining

1. Query in FC is a symbol.
2. Generate all sybmols that can be inferenced (add them to KB) till the query found.
3. Start with known facts (positive literals).
4. If all promises of an implication are known, the its conclusion is added to facts.

Forward Chaining, Cont

```
function FC-ENTAILS?(KB, q) returns true or false

 while agenda is not empty do
  p= POP(agenda)
  unless inferred [p] do
   inferred [p]= true
   foreach c in whose promises p appears do
    Dec (count[c])
    if count[c]= 0 then
     if HEAD[c]=q then
      return true;
     push(HEAD[c],agenda);
 return false
```

Backward chaining

1. Work backward from the query q.

Backward chaining

1. Work backward from the query q.
2. If q is already known, then finished.

Backward chaining

1. Work backward from the query q.
2. If q is already known, then finished.
3. Try to prove all the promising of rules concluding q.

Backward chaining

1. Work backward from the query q.
2. If q is already known, then finished.
3. Try to prove all the promising of rules concluding q.
4. If all promises of an implication are known, the its conclusion is added to facts.

Resolution

1. Every well-formed formula can be represented in Conjunctive Normal Form (CNF).

Resolution

1. Every well-formed formula can be represented in Conjunctive Normal Form (CNF).
2. Resolution:
3. $$\frac{l_1 \vee l_1 \cdots l_k, m_1 \vee m_2 \vee m_n}{l_1 \vee \cdots l_{i-1} \vee l_{i+1} \vee \cdots l_k \vee m_1 \vee \cdots m_{j-1} \vee m_{j+1} \vee \cdots m_n}$$
   where l_i and m_j are complementary literals.

Resolution

1. Every well-formed formula can be represented in Conjunctive Normal Form (CNF).
2. Resolution:
3. $$\frac{l_1 \vee l_1 \cdots l_k, m_1 \vee m_2 \vee m_n}{l_1 \vee \cdots l_{i-1} \vee l_{i+1} \vee \cdots l_k \vee m_1 \vee \cdots m_{j-1} \vee m_{j+1} \vee \cdots m_n}$$
   where l_i and m_j are complementary literals.
4. Resolution is sound and complete for propositional logic.

Resolution, Cont

1. Tries to resolve any two clause until one of the conditions satisfied:

Resolution, Cont

1. Tries to resolve any two clause until one of the conditions satisfied:
2. If no new clause can be added returns false.

Resolution, Cont

1. Tries to resolve any two clause until one of the conditions satisfied:
2. If no new clause can be added returns false.
3. If an empty clause generated, returns true.

Resolution, Cont

```
function Resolution?(KB, q) returns true or false

 clauses= set of clauses in the CNF format of KB^ not a
 loop
  new= {}
  for each Ci, Cj in clauses do
   resolvants=RESOLVE(Ci,Cj);
   if resolvants contains empty cluases then
    return true
   new= new U resolvants
  if new is a subset of clauses return false;
  clauses= clauses U new
```

Resolution, Cont

1. Inference alg. based on resolution uses proof by contradiction.
2. To show $KB \models a$, they tries to proves that $KB \wedge \neg a$ is unsatisfiable.

Resolution, Cont

1. Inference alg. based on resolution uses proof by contradiction.
2. To show $KB \models a$, they tries to proves that $KB \wedge \neg a$ is unsatisfiable.
3. Resolution+ complete search algorithm: complete inference algorithm.

Propositional Inference based on Model Checking

1. We are going to introduce two families of alg. for inference on propositional logic.

Propositional Inference based on Model Checking

1. We are going to introduce two families of alg. for inference on propositional logic.
   - Backtracking search

Propositional Inference based on Model Checking

1. We are going to introduce two families of alg. for inference on propositional logic.
   - Backtracking search
   - Local search

Complete backtracking search (Davis-Putnam)

```
function DPLL(clauses,sybmols,model) return true or false
 if every clauses is true in model return true
 if some clauses is false in model return false
 P,v=FIND-PURE-SYMBOL(symbols,clauses,model)
 if P!=null return DPLL(clauses,symbols-P,model+(P,v))
 P,value= FIND-UNIT-SYMBOL(clauses,model)
 if P!=null return DPLL(clauses,symbols-P,model+(P,v))
 P=FIRST(symbols);rest=REST(symbols);
 return DPLL(clauses,rest,model+(P,true)) or
        DPLL(clauses,rest,model+(P,false))
```

Local-search algorithm

1. Local search algorithms can be applied in satisfiability problem.

Local-search algorithm

1. Local search algorithms can be applied in satisfiability problem.
2. We need a good evaluation function.

Local-search algorithm

1. Local search algorithms can be applied in satisfiability problem.
2. We need a good evaluation function.
3. Counting number of unsatisfied clauses can be an evaluation function.

Local-search algorithm

1. Local search algorithms can be applied in satisfiability problem.
2. We need a good evaluation function.
3. Counting number of unsatisfied clauses can be an evaluation function.
4. Unfortunatly, min-conflict has many local minima.

Local-search algorithm

1. Local search algorithms can be applied in satisfiability problem.
2. We need a good evaluation function.
3. Counting number of unsatisfied clauses can be an evaluation function.
4. Unfortunatly, min-conflict has many local minima.
5. Various forms of randomness are required.

Local-search algorithm- Cont

```
function WALKSAT(clauses,p,max-flips)returns a model or fai

 model=random assignment of true/false to symbols in clause
 for i:= 1 to max-flip do
  if model satisfies clauses then return model
  clause= a randomly selected clause
                that is false in model.
  with prob. p:
   flip value of randomly selected symbol from clause
  else
   flip symbol in clause which maximizes no. of
                satisfied clauses.
```

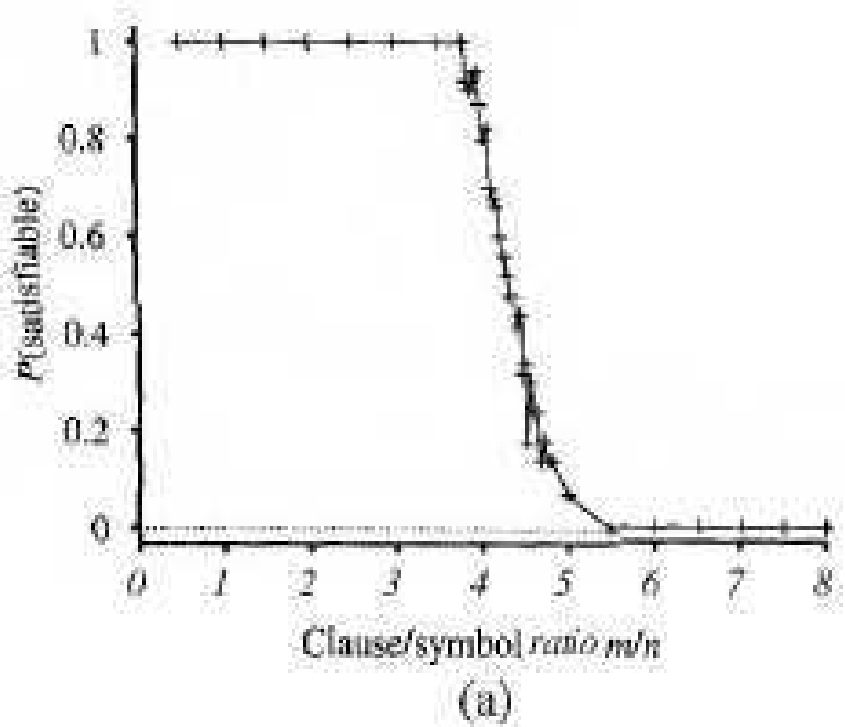
Local-search algorithm, Cont

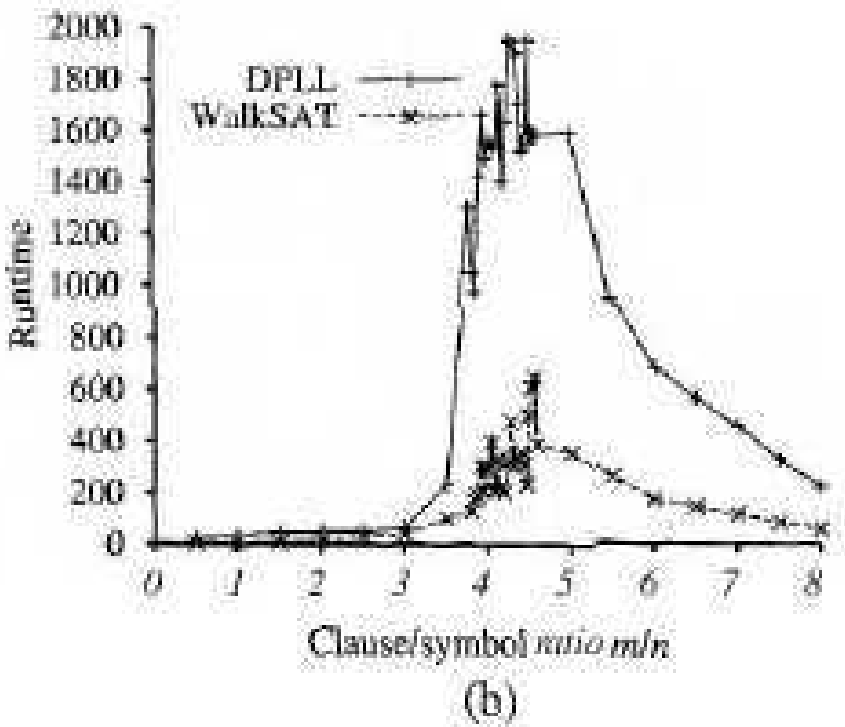
1. WALKSAT may returns a model if one exists.

Local-search algorithm, Cont

1. WALKSAT may returns a model if one exists.
2. It can not tell whether the statement is unsatisfiable.
3. But, WALKSAT can tell us a statement my be unsatisfiable.
4. Satisfiability may be a hard problem (depending on $\frac{No.\ of\ symbols}{No.\ of\ clauses}$)

Local-search algorithm, Cont





An agent for wampus-world

1. $[1, 1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$

3. Stench signals and Wampus relation:

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y})$$

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$
3. Stench signals and Wampus relation:
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y})$$
4. There is exactly one wampus at the begining:

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$
3. Stench signals and Wampus relation:
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y})$$
4. There is exactly one wampus at the begining:
$$W_{1,1} \vee W_{1,2} \vee \cdots W_{4,4}$$

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$
3. Stench signals and Wampus relation:
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y})$$
4. There is exactly one wampus at the begining:
$$W_{1,1} \vee W_{1,2} \vee \cdots W_{4,4}$$
$$\neg W_{x_1,y_1} \vee \neg W_{x_2,y_2}$$

An agent for wampus-world

1. $[1,1]$ is safe: $\neg P_{1,1}$ and $\neg W_{1,1}$.
2. Breeze signals and Pits relation:
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y})$$
3. Stench signals and Wampus relation:
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y})$$
4. There is exactly one wampus at the begining:
$$W_{1,1} \vee W_{1,2} \vee \cdots W_{4,4}$$
$$\neg W_{x_1,y_1} \vee \neg W_{x_2,y_2}$$
5. Problem has 64 symbols and 155 statements.

An agent for wampus-world, Cont

1. Our agent tries to find a provably safe place(i.e. $\neg P_{i,j} and \neg W_{i,j}$).

An agent for wampus-world, Cont

1. Our agent tries to find a provably safe place(i.e. $\neg P_{i,j} and \neg W_{i,j}$).
2. Then, it tries to find a possibly safe place (i.e. $P_{i,j} \vee W_{i,j}$ can not be entailed).

An agent for wampus-world, Cont

1. Our agent tries to find a provably safe place(i.e. $\neg P_{i,j} and \neg W_{i,j}$).
2. Then, it tries to find a possibly safe place (i.e. $P_{i,j} \vee W_{i,j}$ can not be entailed).
3.