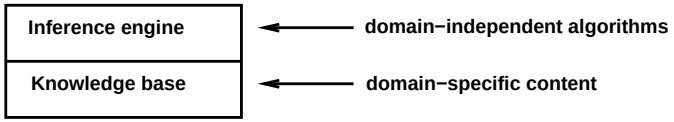
LOGICAL AGENTS

Introduction

- So far we have looked mainly at how agents can use techniques based on search to decide what they might do.
- This is one major area of artificial intelligence.
- Today we will start to look at another major area, the use of *knowledge* about the world, and *reasoning* that we might do with the knowledge.
- In particular we will look at the use of *logic* which has been widely used in AI and used for a long time.

Knowledge bases



- *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

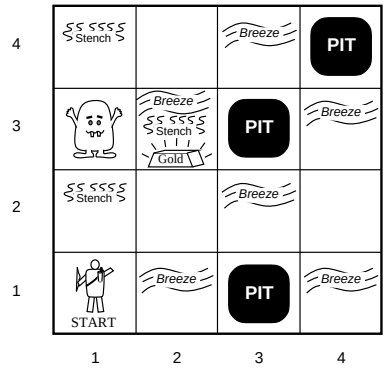
- Agents can be viewed at the *knowledge level*
  - *what they know*, regardless of how implemented
- Or at the *implementation level*
  - data structures in KB and algorithms that manipulate them

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*

- An agent has to 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

A gaming context

Wumpus world



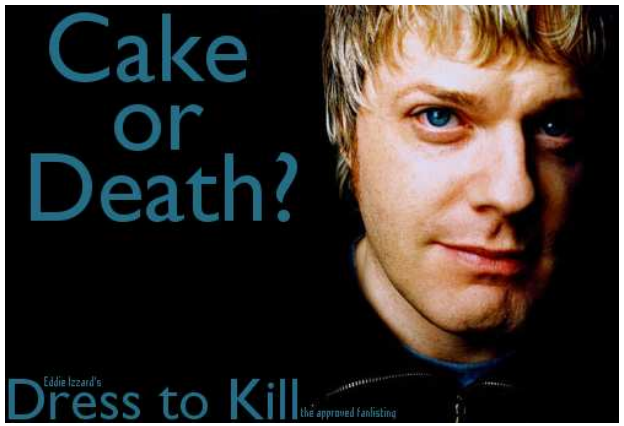
- *Actuators:* Left turn, Right turn Forward, Grab, Release, Shoot
- *Sensors:* Breeze, Glitter, Smell

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

Performance measure

- Gold
  - Score: +1000
- Death
  - Score: -1000
- Taking a step
  - Score: -1
- Use of arrow
  - Score: -10



Characterization of WW

- *Observable/Accessible?*

Characterization of WW

- *Observable/Accessible?* No—only *local* perception

Characterization of WW

- *Observable/Accessible?* No—only *local* perception
- *Deterministic?*

Characterization of WW

- *Observable/Accessible?* No—only *local* perception
- *Deterministic* Yes—outcomes exactly specified

Characterization of WW

- *Observable/Accessible?* No—only *local* perception
- *Deterministic* Yes—outcomes exactly specified
- *Episodic?*

Characterization of WW

- *Observable/Accessible?* No—only *local* perception
- *Deterministic* Yes—outcomes exactly specified
- *Episodic* No—sequential at the level of actions

Characterization of WW

- *Observable/Accessible?* No—only *local* perception
- *Deterministic* Yes—outcomes exactly specified
- *Episodic* No—sequential at the level of actions
- *Static?*

Characterization of WW

- *Observable/Accessible?* 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

Characterization of WW

- *Observable/Accessible?* 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?*

Characterization of WW

- *Observable/Accessible?* 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

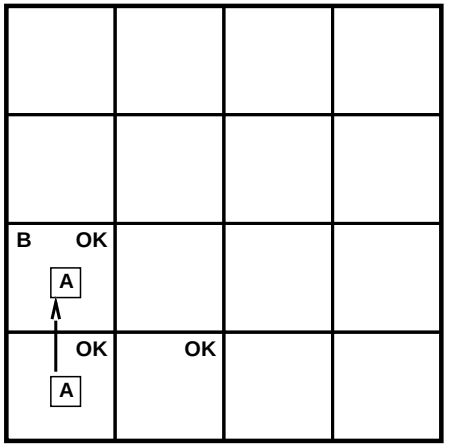
Characterization of WW

- *Observable/Accessible?* 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?*

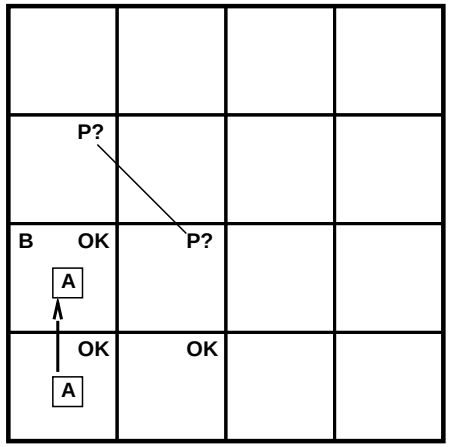
Characterization of WW

- *Observable/Accessible?* 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

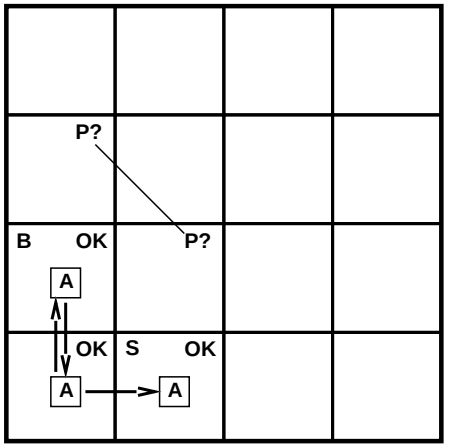
Exploring the WW



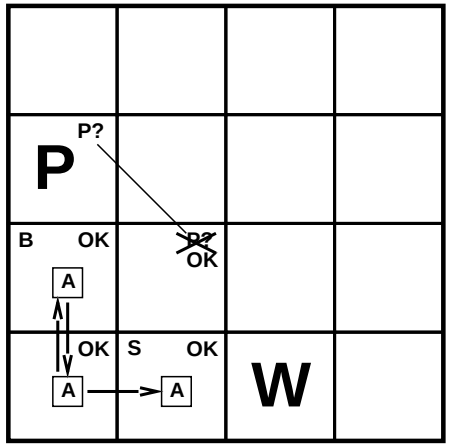
Exploring the WW



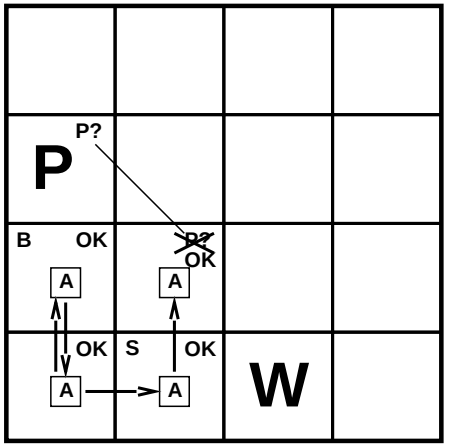
Exploring the WW



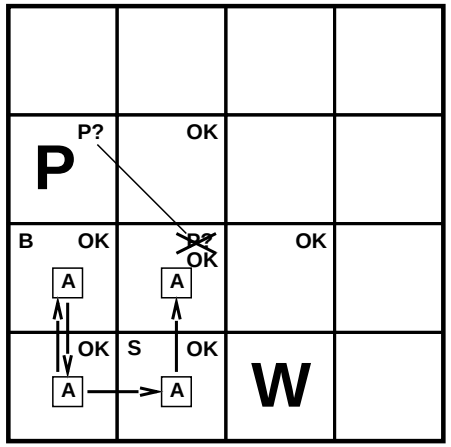
Exploring the WW



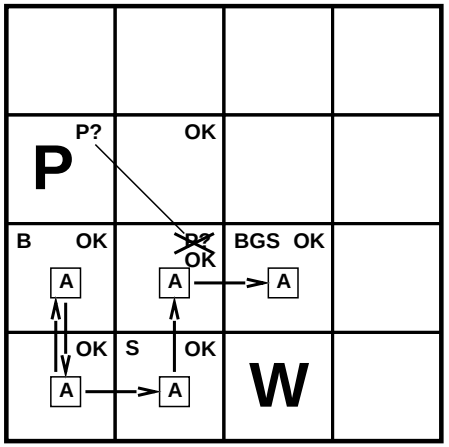
Exploring the WW



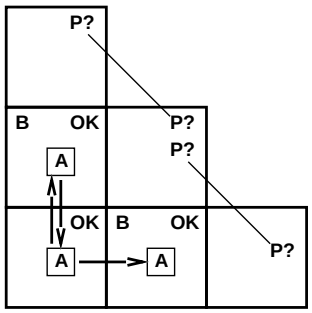
Exploring the WW



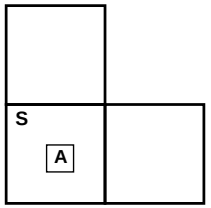
Exploring the WW



Other tight spots



- Breeze in (1,2) and (2,1) $\Rightarrow$ no safe actions
- Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31



- Smell in (1,1) $\Rightarrow$ cannot move
- Can use a strategy of *coercion*:
  - Shoot straight ahead
  - Wumpus was there $\Rightarrow$ dead $\Rightarrow$ safe
  - Wumpus wasn't there $\Rightarrow$ safe

Logic

Logic really

- *Logics* are formal languages for representing information such that conclusions can be drawn
- *Syntax* defines the sentences in the language
- *Semantics* define the "meaning" of sentences
  - Define *truth* of a sentence in a world
- For example, 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 $x = 0,\ y = 6$

Entailment

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

$$KB \models \alpha$$

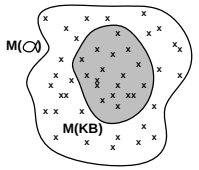
- Knowledge base *KB* entails sentence α *if and only if* α is true in all worlds where *KB* is true.
- The KB containing "the Giants won" and "the Jets won" entails "The Giants won or the Jets won"
- $x + y = 4$ entails $4 = x + y$
- Entailment is a relationship between sentences, *syntax*, that is based on *semantics*

Model

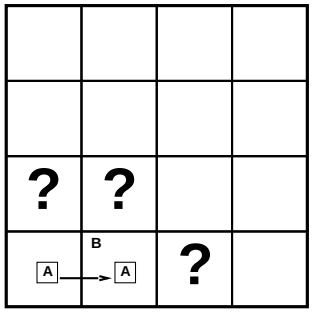
- Logicians typically think in terms of *models*, which are formally structured worlds with respect to which truth can be evaluated
- We say *m is a model of* a sentence α if α is true in *m*
- $M(\alpha)$ is the set of all models of α
- Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

KB = Giants won and Jets won

α = Jets won

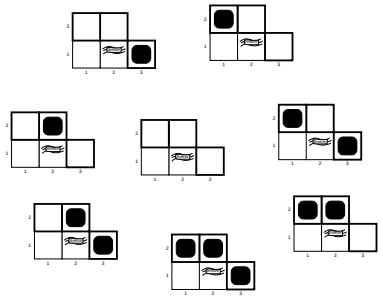


Entailment in WW

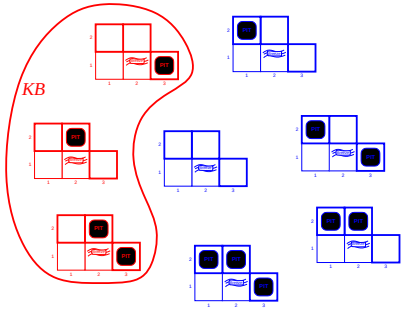


- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models, assuming only pits
- 3 Boolean choices $\Rightarrow$ 8 possible models

WW Models

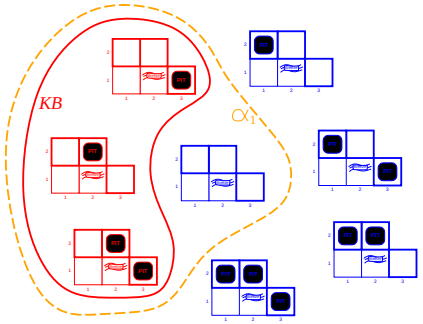


WW Models



KB = wumpus-world rules + observations

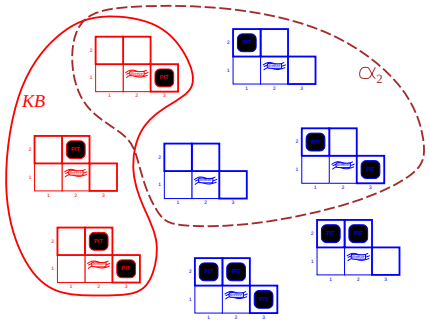
WW Models



KB = wumpus-world rules + observations

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

WW Models



KB = wumpus-world rules + observations

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

Inference

- When:

$$KB \models \alpha$$

We say that α is a *logical consequence* of *KB*.

- Consequences of *KB* are a haystack

  α is a needle.

- $KB \vdash_i \alpha$ says sentence α can be derived from *KB* by some *inference procedure* *i*.
- If the KB entails α there is a needle in haystack

  Inference tells us how to find it.

- *Soundness*: *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$.

  If we find the needle, it is good

  If our inference procedure finds α, the it is a logical consequence.

- *Completeness* *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

  We can find every good needle

  Our inference procedure will find every logical consequence.

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*.
- But first we will look at a simpler logic.
- (This is a subset of first order logic, so we will reuse everything that we do here.)

Propositional Logic — Syntax

- Propositional logic is the simplest logic—illustrates basic ideas
- **Definition:** A *proposition* is a statement that can be either *true* or *false*; it must be one or the other, and it cannot be both.
- The following are propositions:
  - the reactor is on;
  - the wing-flaps are up;
  - Marvin K Mooney is president.

  whereas the following are not:
  - are you going out somewhere?
  - 2+3
- A good test for a proposition is to ask "Is it true that. . . ?".
- If that makes sense, it is a proposition.

- Now, rather than write out propositions in full, we will abbreviate them by using *propositional variables*.
- It is standard practice to use the lower-case roman letters

  $$p, q, r, \ldots$$

  to stand for propositions.
- Just to be confusing, we, like the textbook have been using the proposition symbols $B_{1,2}$ to stand for:

  It is breezy in [1, 2].
- These propositions, which indicate a single thing in the real world, are called *atomic propositions*.
- These propositions are also a simple form of *sentence*.
- We can also construct more complex sentences in the form of *compound propositions*.

- If S is a sentence, $\neg S$ is a sentence
  negation
- If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence
  conjunction
- If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence
  disjunction
- If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence
  implication
- If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence
  biconditional
- Given some *language* (set of propositions), we will write $\mathcal{W}$ to denote *all* the sentences that can be constructed using these rules.

Propositional logic — Semantics

- Each model specifies true/false for each proposition symbol
- Also called an *interpretation*
- Also called a *valuation*
- Here's a model/interpretation/valuation:

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
true	*true*	*false*

- With these symbols, 8 possible models/interpretations/valuations, can be enumerated automatically

- 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	
$S_1 \Rightarrow S_2$	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	$S_1 \Rightarrow S_2$	is true *and*	$S_2 \Rightarrow S_1$	is true	

- Simple recursive process evaluates an arbitrary sentence

  $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{false} \vee \mathit{true}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$

Truth tables for connectives

- These rules correspond to the truth tables for the 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*

- Each line in a truth table is a model.

Validity and satisfiability

- If we consider all possible models, there are different properties that may hold for a sentence.
- A sentence is *valid* if it is true in *all* models.

  True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Validity is connected to inference via the *Deduction Theorem*:

  $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid
- A sentence that is *valid* is said to be a *tautology*.

- A sentence is *satisfiable* if it is true in *some* model

  $A \vee B$ C
- It is satisfiable if it is possible to make it true by picking the right truth values for its atomic propositions.
- If a sentence is satisfiable we also say that it is *consistent*.

- A sentence is *unsatisfiable* if it is true in *no* models

  $A \wedge \neg A$
- If a sentence is unsatisfiable, then clearly it is not satisfiable (hence the name).
- It is clearly not consistent either, so we call it *inconsistent*.
- Satisfiability is connected to inference via:

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

  Thus we can prove α by *reductio ad absurdum*

Truth tables for inference

- Recall that $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

  We illustrated this graphically for the WW.
- Turns out we can determine if $M(KB) \subseteq M(\alpha)$ using truth tables.
- We write out every possible combination of truth values for the atomic propositions.

  Enumerate all the models.
- If *KB* is true in row, check that α is too.

  If this is the case, then $M(KB) \subseteq M(\alpha)$ and so $KB \models \alpha$

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*

- Here $KB \models B_{2,1}$.

Algorithm for this inference procedure

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))
```


- TT-CHECK-ALL is first called with an empty model and recursively expands it.
  - Tries to avoid enumerating all the models
- PL-TRUE? returns true if a sentence holds in the model.
- EXTEND(P, *true*, *model*)) extends a partial model (only gives values for some propositions) with one in which P is true.
- This algorithm is *sound* and *complete*
- For n symbols this is $O(2^n)$.
  - co-NP-complete
- So, that is not good, but it does give our logical agent a way to figure things out about the world.

- Here is another way to think about what we are doing here.
- **Theorem:**

$$\{S_1, \ldots, S_n\} \models C$$

iff

$$\models (S_1 \wedge \cdots \wedge S_n) \Rightarrow C$$

- So we have a method for determining whether C is a logical consequence of $S_1, \ldots S_n$.
- We use a truth table to see whether $S_1 \wedge \cdots \wedge S_n \Rightarrow C$ is a tautology.
- If it is, then C is a logical consequence of $S_1, \ldots, S_n$.
- We call this the *truth table method*.

- For example, to show that

$$p \wedge q \models p \vee q.$$

To do this, we construct a truth-table for

$$(p \wedge q) \Rightarrow (p \vee q).$$

Here it is:

		(1)	(2)	
p	q	$p \wedge q$	$p \vee q$	$(1) \Rightarrow (2)$
false	*false*	*false*	*false*	*true*
false	*true*	*false*	*true*	*true*
true	*false*	*false*	*true*	*true*
true	*true*	*true*	*true*	*true*

Since

$$(p \wedge q) \Rightarrow (p \vee q).$$

is true in every model, we have that $p \vee q$ is a logical consequence of $p \wedge q$.

Logical equivalence

- Two sentences are emphlogically equivalent iff they are true in same models.
- $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$
- There are some common equivalences on the next slide.
- These are universal tautologies — they will always be true in all possible models.
- These are handy because they are tautologies, and so we can bring them into proofs (when we get to that).

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

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
  - heuristic search in model space (sound but incomplete)
- *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.
  - May require translation of sentences into a *normal form*

'Syntactic' Proof

- The idea of syntactic proof is to replace the checking of models to determine whether a formula is valid by a procedure that involves purely *syntactic* manipulation.
- The kinds of techniques that we shall use are similar to those that we use when solving problems in algebra.
- The basic idea is that to show that C is a logical consequence of $S_1, \ldots, S_n$, we use a set of *rules* to manipulate formulae.

  If we can derive C from $S_1, \ldots, S_n$ by using these rules, then C is said to be *proved* from $S_1, \ldots, S_n$, which we indicate by writing

$$S_1, \ldots, S_n \vdash C.$$

- The symbol $\vdash$ is called the *syntactic turnstile*.
- An expression of the form

$$S_1, \ldots, S_n \vdash C$$

is called a *syntactic sequent*.

- A rule has the general form:

$$\frac{\vdash S_1; \cdots; \vdash S_n}{\vdash S} \text{ rule name}$$

Such a rule is read:

If
$S_1, \ldots, S_n$ are proved
then
S is proved.

- Here is an example of such a rule:

$$\frac{\vdash S_1; \vdash S_2}{\vdash S_1 \wedge S_2} \wedge\text{-I}$$

- This rule is called *and introduction*. It says that if we have proved S_1, and we have also proved S_2, then we can prove $S_1 \wedge S_2$.
- This should remind you a lot of the truth table for $\wedge$

- Here is another rule:

$$\frac{\vdash S_1 \wedge S_2}{\vdash S_1; \vdash S_2} \wedge\text{-E}$$

- This rule is called *and elimination*. It says that if we have proved $S_1 \wedge S_2$, then we can prove both S_1 and S_2.
- It allows us to eliminate the $\wedge$ symbol from between them.

- Let us now try to define precisely what we mean by *proof*.
- **Definition:** (Proof) If

$$\{S_1, \ldots, S_n, C\} \subseteq \mathcal{W}$$

then there is a proof of C from $S_1, \ldots, S_n$ iff there exists some sequence of formulae

$$T_1, \ldots, T_m$$

such that $T_m = C$, and each formula T_k, for $1 \leq k < m$ is either one of the formula $S_1, \ldots, S_n$, or else is the conclusion of a rule whose antecedents appeared earlier in the sequence.

The sequence of formulae $T_1, \ldots, T_m$ is the *proof*.

- If there is a proof of C from $S_1, \ldots, S_n$, then we indicate this by writing:

$$S_1, \ldots, S_n \vdash C$$

- It should be clear that the symbols $\vdash$ and $\models$ are related. We now have to state exactly *how* they are related.
- There are two properties of $\vdash$ to consider:
  - *soundness*;
  - *completeness*.
- Intuitively, $\vdash$ is said to be *sound* if it is correct, in that it does not let us derive something that is not true.
- Intuitively, *completeness* means that $\vdash$ will let us prove anything that is true.

- **Definition:** (Soundness) A proof system $\vdash$ is said to be *sound* with respect to semantics $\models$ iff

$$S_1, \ldots, S_n \vdash C$$

implies

$$S_1, \ldots, S_n \models C.$$

- **Definition:** (Completeness) A proof system $\vdash$ is said to be *complete* with respect to semantics $\models$ iff

$$S_1, \ldots, S_n \models C$$

implies

$$S_1, \ldots, S_n \vdash C$$

Natural deduction

- There are many proof systems for propositional logic; we shall look at a simple one.
  - Natural deduction
- First, we have an unusual rule that allows us to introduce any tautology.

$$\frac{}{\vdash S}\ \text{TAUT}\quad \text{if } S \text{ is a tautology}$$

- Because a tautology is true there is no problem bringing it into the proof.
- An example tautology is "Either the Mets won, or they lost".

- Next, rules for *eliminating* connectives.

$$\frac{\vdash S_1 \wedge S_2}{\vdash S_1; \vdash S_2}\ \wedge\text{-E}$$

- We already saw this one:

  If we are told "The Jets won and the Giants won", then we know "The Jets won" and we know "The Giants won".

- Or-elimination

$$\frac{\begin{array}{l}\vdash S_1 \vee \cdots \vee S_n; \\ S_1 \vdash C; \\ \quad \cdots; \\ S_n \vdash C\end{array}}{\vdash C}\ \vee\text{-E}$$

- This is more complex.

  I hear on the radio that "New York won today", so I know "The Jets won, or the Giants won, or both". Now I have bet on both teams to win, so "If the Jets won, I have won some money" and "If the Giants won, I won some money", so, overall "I won some money".

- An alternative version of the $\vee$ elimination rule is:

$$\frac{\begin{array}{l}\vdash S_1 \vee S_2; \\ \vdash S_1 \Rightarrow C; \\ \vdash S_2 \Rightarrow C\end{array}}{\vdash C}\ \vee\text{-E}$$

- Next, a rule called *modus ponens*, which lets us eliminate $\Rightarrow$.

$$\frac{\vdash S_1 \Rightarrow S_2; \vdash S_1}{\vdash S_2} \Rightarrow\text{-E}$$

- We use that here:

  If I know "If the Jets won, they qualified for the playoffs", and I learn "The Jets won", the I can conclude "The Jets qualified for the playoffs"

- Next, rules for *introducing* connectives.

$$\frac{\vdash S_1; \cdots; \vdash S_n}{\vdash S_1 \wedge \cdots \wedge S_n} \wedge\text{-I}$$

- Example:

  "The Jets won" and "The Giants won", so "The Jets and the Giants won".

- Here is Or-introduction:

$$\frac{\vdash S_1}{\vdash S_1 \vee \cdots \vee S_n} \vee\text{-I}$$

- Example:

  "The Jets won" so "The Jets or the Giants won".

- We have a rule called the *deduction theorem*. This rule says that if we can prove S_2 from S_1, then we can prove that $S_1 \Rightarrow S_2$.

$$\frac{S_1 \vdash S_2}{S_1 \Rightarrow S_2} \Rightarrow\text{-I}$$

- So, what is an example?

  If somebody tells me "The Jets won", and then I realise after thinking about my bets, that "I have won some money", then I can conclude that "If the Jets win, I will win some money".

- There are a whole range of other rules, which we shall not list here.
- I'll post a handout with them on the course web-page.
- Rather than go through them, let's look at what a proof is.

- Recall that a proof is defined as:

$$S_1, \ldots, S_n \vdash C$$

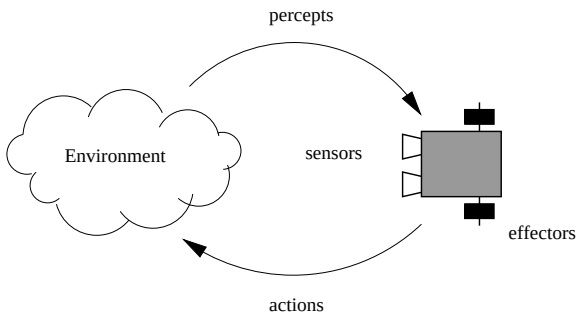
iff there exists some sequence of formulae

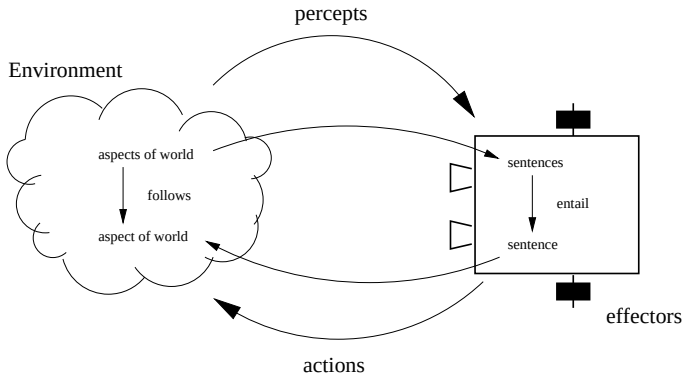
$$S_1, \ldots, S_m$$

such that $S_m = C$, and each formula S_k, for $1 \leq k < m$ is either one of the formula $S_1, \ldots, S_n$, or else is the conclusion of a rule whose antecedents appeared earlier in the sequence.

- The sequence of formulae $S_1, \ldots, S_m$ is the *proof*.

Remind me why this is important?





Proof Examples

- Example 1:
  $p \wedge q \vdash q \wedge p$

1.	$p \wedge q$	Given
2.	p	From 1 using $\wedge$-E
3.	q	1,$\wedge$-E
4.	$q \wedge p$	2, 3, $\wedge$-I

- If that seems simple, that is because it is. Don't worry, proofs get more complex than this.

- Example 2:
  $p \wedge q \vdash p \vee q$

1.	$p \wedge q$	Given
2.	p	1, $\wedge$-E
3.	$p \vee q$	2, $\vee$-I

- Another simple one.

- Example 3:
  $p \wedge q, p \Rightarrow r \vdash r$

1.	$p \wedge q$	Given
2.	p	1, $\wedge$-E
3.	$p \Rightarrow r$	Given
4.	r	2, 3, $\Rightarrow$-E

- Here we use modus ponens.

- Example 4:
  $p \Rightarrow q, q \Rightarrow r \vdash p \Rightarrow r$

1.	$p \Rightarrow q$	Given	
2.	$q \Rightarrow r$	Given	
3.	p	As.	|
4.	q	1, 3, $\Rightarrow$-E	|
5.	r	2, 4, $\Rightarrow$-E	|
6.	$p \Rightarrow r$	3, 5, $\Rightarrow$-I	

- Here we make an assumption and then *discharge* it at the end.

- Example 5:
  $(p \wedge q) \Rightarrow r \vdash p \Rightarrow (q \Rightarrow r)$

1.	$(p \wedge q) \Rightarrow r$	Given	
2.	p	As.	|
3.	q	As.	||
4.	$p \wedge q$	2, 3, $\wedge$-I	||
5.	r	1, 4, $\Rightarrow$-I	||
6.	$q \Rightarrow r$	3–5, $\Rightarrow$-I	|
7.	$p \Rightarrow (q \Rightarrow r)$	2–6, $\Rightarrow$-I	

- We can make more than one assumption, but we need to discharge them all.

- Example 6:
  $p \Rightarrow (q \Rightarrow r) \vdash (p \wedge q) \Rightarrow r$

1.	$p \Rightarrow (q \Rightarrow r)$	Given	
2.	$p \wedge q$	As.	|
3.	p	2, $\wedge$-E	|
4.	q	2, $\wedge$-E	|
5.	$q \Rightarrow r$	1, 3, $\Rightarrow$-E	|
6.	r	4, 5, $\Rightarrow$-E	|
7.	$(p \wedge q) \Rightarrow r$	2–6, $\Rightarrow$-I	

- Example 7:
  $p \Rightarrow q, \neg q \vdash \neg p$

1.	$p \Rightarrow q$	Given	
2.	$\neg q$	Given	
3.	p	As.	|
4.	q	1, 3, $\Rightarrow$-E	|
5.	$q \wedge \neg q$	2, 4, $\wedge$-I	|
6.	$\neg p$	3, 5, $\neg$-I	

- Here discharging the assumption needs a rule we didn't look at before.

- The rule allows us to negate an assumption it it leads us to a contradiction.

$$\frac{S \vdash \bot}{\vdash \neg S}\ \neg\text{-I}$$

- $\bot$ stands for any formula which is unsatisifable, for example

$$S \wedge \neg S$$

- We call such a formula a *contradiction* and say that it is *inconsistent*.
- The notion behind the rule is that if we assume something, and that leads us to an impossible conclusion, what we assumed has to be wrong.

- Example 8:
  $p \Rightarrow q \vdash \neg(p \wedge \neg q)$

1.	$p \Rightarrow q$	Given	
2.	$p \wedge \neg q$	As.	|
3.	p	2, $\wedge$-E	|
4.	$\neg q$	2, $\wedge$-E	|
5.	q	1, 3, $\Rightarrow$-E	|
6.	$q \wedge \neg q$	4, 5, $\wedge$-I	|
7.	$\neg(p \wedge \neg q)$	6, $\neg$-I	

- This is another 'proof by contradiction" or (to be fancy) *reductio ad absurdum*.

- Example 9:

  Can Jim will party all night and pass AI?
  That must be wrong.
  If he works hard he won't have time to party. If he doesn't work hard he's not going to pass AI.

  Let:

  p Jim will party all night
  q Jim will pass AI
  r Jim works hard

  Formalisation of argument:

$$r \Rightarrow \neg p, \neg r \Rightarrow \neg q \vdash \neg(p \wedge q)$$

- Can we prove that it is not possible for Jim to party and pass AI ($\neg(p \wedge q)$)?

1.	$r \Rightarrow \neg p$	Given	
2.	$\neg r \Rightarrow \neg q$	Given	
3.	$p \wedge q$	As.	|
4.	r	As.	||
5.	$\neg p$	1, 4, $\Rightarrow$-E	||
6.	p	3, $\wedge$-I	||
7.	$p \wedge \neg p$	5, 6, $\wedge$-I	||
8.	$\neg r$	4, 7, $\neg$-I	|
9.	$\neg q$	2, 9, $\Rightarrow$-E	|
10.	q	3, $\wedge$-E	|
11.	$q \wedge \neg q$	9, 10, $\wedge$-I	|
12.	$\neg(p \wedge q)$	3, 11, $\neg$-I	

Summary

- This lecture covered an introduction to using logic to program agents.
- We started with some motivation — having an agent figure out some facts in the Wumpus world.
- We showed informally how this might be done using models.
- We then introdcued propositional logic, a formal system for reasoning, and showed how we can automate the model checking process.
- Finally we looked at approaches to proof based on symbol manipulation.