The Big Idea

• Humans know stuff

• We use the stuff we know to help us do things
The Big Idea

• Do our agents know stuff?
The Big Idea

• Do our agents know stuff?
  – Well, kind of…
The Big Idea

• Do our agents know stuff?
  – Well, kind of…

• Knowledge encoded in agent functions:
The Big Idea

• Do our agents know stuff?
  – Well, kind of…

• Knowledge encoded in agent functions:
  – Successor functions
  – Heuristics
  – Performance measures
  – Goal tests
The Big Idea

• Think about n-puzzle agent
  – Can it predict outcomes of future actions?
  – Can it conclude that a state is unreachable?
  – Can it prove that certain states are always unreachable from others?

• How to represent an environment that is
  – Atomic
  – Partially observable
The Big Idea

- Agents we’ve designed so far possess very inflexible knowledge
- What if we could teach our agents how to reason?
  - Combine information
  - Adapt to new tasks
  - Learn about environment
  - Update in response to environmental changes
- Agent will need a way to keep track of and apply stuff it knows
Outline

• Knowledge-based agents
• Wumpus world
• Logic in general - models and entailment
• Propositional (Boolean) logic
• Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – Backward chaining
  – Resolution
Outline

• Knowledge-based agents
• Wumpus world
• Logic in general - models and entailment
• Propositional (Boolean) logic
• Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – Backward chaining
  – Resolution
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
  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
Knowledge Based Agents

• When knowledge-based agent runs it:
  – Tells KB about its latest perception
    • MAKE-PERCEPT-SENTENCE
  – Asks the KB what to do next
    • MAKE-ACTION-QUERY
  – Executes action and tells the KB so
    • MAKE-ACTION-SENTENCE
A Simple Knowledge Based Agent

function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t← t + 1
    return action

• The agent must be able to:
A Simple Knowledge Based Agent

The agent must be able to:

- Represent states, actions, etc.

```
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
```
A Simple Knowledge Based Agent

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts

```plaintext
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
```
A Simple Knowledge Based Agent

- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world

```python
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
```
A Simple Knowledge Based Agent

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

```java
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
```
A Simple Knowledge Based Agent

function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
    t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action

• The agent must be able to:
  – Represent states, actions, etc.
  – Incorporate new percepts
  – Update internal representations of the world
  – Deduce hidden properties of the world
  – Deduce appropriate actions
Outline

• Knowledge-based agents
• **Wumpus world**
• Logic in general - models and entailment
• Propositional (Boolean) logic
• Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – Backward chaining
  – Resolution
Wumpus World PEAS Description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
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
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
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:** Stench, Breeze, Glitter, Bump, Scream
Wumpus World Characterization

• Fully Observable
Wumpus World Characterization

- **Fully Observable** No – only local perception
Wumpus World Characterization

- **Fully Observable** No – only local perception
- **Deterministic**
Wumpus World Characterization

- **Fully Observable** No – only local perception
- **Deterministic** Yes – outcomes exactly specified
Wumpus World Characterization

- **Fully Observable** No – only *local* perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic**
Wumpus World Characterization

• **Fully Observable** No – only *local* perception
• **Deterministic** Yes – outcomes exactly specified
• **Episodic** No – sequential at the level of actions
Wumpus World Characterization

- **Fully Observable** No – only local perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic** No – sequential at the level of actions
- **Static**
Wumpus World Characterization

- **Fully 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
Wumpus World Characterization

- **Fully 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

- **Fully 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

- **Fully 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

- **Fully 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
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Outline

• Knowledge-based agents
• Wumpus world
• **Logic in general - models and entailment**
  • Propositional (Boolean) logic
  • Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – Backward chaining
  – Resolution
Logic in General

- **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;  
  - i.e., define **truth** of a sentence in a world
Logic in General

- **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;
  - i.e., define truth of a sentence in a world.

- E.g., the language of arithmetic
  - $x + 2 \geq y$ is a sentence; $x + 2 + y > \emptyset$ 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 $x = 0, y = 6$.
  - True in all worlds?
  - False in all worlds?
Entailment

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

\[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true
  - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)
Entailment

- **Entailment** means that one thing follows from another
  
  \[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true
  - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)

- Which part is syntax and which part is semantics?
Entailment

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

\[ \text{KB } \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true
  - E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)

- Which part is syntax and which part is semantics?
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
Models

- 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 $\alpha$ if $\alpha$ is true in $m$.
  - Alternative phrasing: $m$ satisfies $\alpha$.

- $M(\alpha)$ is the set of all models of $\alpha$.
  - All models where $\alpha$ is true.

- Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$.
  - E.g. $KB = \text{Giants won and Reds won}$, $\alpha = \text{Giants won}$.
Entailment in the Wumpus World

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

Consider possible models for $KB$ assuming only pits

? Boolean choices
Entailment in the Wumpus World

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

Consider possible models for $KB$ assuming only pits

3 Boolean choices $\Rightarrow$ ? possible models
Entailment in the Wumpus World

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

Consider possible models for $KB$ assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models
Wumpus Models
Wumpus Models

- $KB = \text{wumpus-world rules} + \text{observations}$
Wumpus Models

- $KB = \text{wumpus-world rules} + \text{observations}$
- $\alpha_1 = \text{"[1,2] is safe"}, \ KB \models \alpha_1$, proved by model checking
Wumpus Models

- $KB = \text{wumpus-world rules} + \text{observations}$
Wumpus Models

- $KB = \text{wumpus-world rules} + \text{observations}$
- $\alpha_2 = "[2,2] \text{ is safe}"$, $KB \not\models \alpha_2$
Inference

• KB is your agent’s haystack
  – Pile containing all possible conclusions
• Specific conclusion $\alpha$ is the needle agent is looking for
• Entailment: Is the needle in the haystack?
• Inference: Can you find the needle?
Inference

- $KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } I$
  - $\alpha$ is derived from KB by I
  - $i$ derives $\alpha$ from KB
- **Soundness:** $i$ is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
  - a.k.a. truth-preserving
Inference

- $KB \vdash_i \alpha$ = sentence $\alpha$ can be derived from $KB$ by procedure $I$
  - $\alpha$ is derived from $KB$ by $I$
  - $i$ derives $\alpha$ from $KB$
- **Soundness:** $i$ is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
  - a.k.a. truth-preserving
- Is model checking sound?
Inference

- \( KB \models_i \alpha = \) sentence \( \alpha \) can be derived from \( KB \) by procedure \( I \)
  - \( \alpha \) is derived from \( KB \) by \( I \)
  - \( I \) derives \( \alpha \) from \( KB \)

- **Soundness**: \( I \) is sound if whenever \( KB \models_i \alpha \), it is also true that \( KB \models \alpha \)
  - a.k.a. truth-preserving

- Is model checking sound? **Yes**
Inference

- \( KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } I \)
  - \( \alpha \) is derived from KB by I
  - \( i \) derives \( \alpha \) from KB
- **Soundness**: \( i \) is sound if whenever \( KB \vdash_i \alpha \), it is also true that \( KB \models \alpha \)
  - a.k.a. truth-preserving
- Is model checking sound? **Yes**
- What would it mean for an inference algorithm to *not* be sound?
Inference

- \( KB \models_i \alpha \) = sentence \( \alpha \) can be derived from \( KB \) by procedure \( I \)
  - \( \alpha \) is derived from \( KB \) by \( I \)
  - \( i \) derives \( \alpha \) from \( KB \)
- **Soundness**: \( i \) is sound if whenever \( KB \models_i \alpha \), it is also true that \( KB \models \alpha \)
  - a.k.a. truth-preserving
- **Completeness**: \( i \) is complete if whenever \( KB \models \alpha \), it is also true that \( KB \models_i \alpha \)
- 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 \( KB \).
Outline

• Knowledge-based agents
• Wumpus world
• Logic in general - models and entailment
• Propositional (Boolean) logic
• Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – Backward chaining
  – Resolution
Propositional Logic: Syntax

• Propositional logic is the simplest logic
  – Illustrates basic ideas
• The proposition symbols $S_1$, $S_2$ etc are sentences
• Atomic sentences: Single proposition
  – Special fixed meaning symbols: True and False
• Complex sentences:
  – If $S$ is a sentence, $\neg S$ is a sentence (negation)
  – If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction)
  – If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ 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)
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$, $P_{2,2}$, $P_{3,1}$

false true false

With these symbols, 8 possible models, can be enumerated automatically. Rules for evaluating truth with respect to a model $m$: 
## Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

<table>
<thead>
<tr>
<th></th>
<th>$P_{1,2}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

$\neg S$ is true iff
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:
$$\neg S$$ is true iff  $S$ is false
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \) \( \quad P_{2,2} \quad P_{3,1} \)
false \quad true \quad false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\[ \neg S \text{ is true iff } S \text{ is false} \]
\[ S_1 \land S_2 \text{ is true iff } \]
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

- $\neg S$ is true iff $S$ is false
- $S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \), \( P_{2,2} \), \( P_{3,1} \)
false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\[ \neg S \text{ is true iff } S \text{ is false} \]

\[ S_1 \land S_2 \text{ is true iff } S_1 \text{ is true} \text{ and } S_2 \text{ is true} \]

\[ S_1 \lor S_2 \text{ is true iff } S_1 \text{ is true or } S_2 \text{ is true} \]
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

$\neg S$ is true iff $S$ is false

$S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true

$S_1 \lor S_2$ is true iff $S_1$ is true or $S_2$ is true
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \) \( P_{2,2} \) \( P_{3,1} \)

false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\[ \neg S \text{ is true iff } S \text{ is false} \]

\[ S_1 \land S_2 \text{ is true iff } S_1 \text{ is true and } S_2 \text{ is true} \]

\[ S_1 \lor S_2 \text{ is true iff } S_1 \text{ is true or } S_2 \text{ is true} \]

\[ S_1 \Rightarrow S_2 \text{ is true iff } \]
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

- $\neg S$ is true iff $S$ is false
- $S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true
- $S_1 \lor S_2$ 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
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \), \( P_{2,2} \), \( P_{3,1} \)

false, true, false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\[-S\] is true iff \( S \) is false

\( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true

\( S_1 \lor S_2 \) 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
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

- $\neg S$ is true iff $S$ is false
- $S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true
- $S_1 \lor S_2$ 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
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \) \( P_{2,2} \) \( P_{3,1} \)

false \( \) true \( \) false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\( \neg S \) is true iff \( S \) is false

\( S_1 \land S_2 \) is true iff \( S_1 \) is true \text{ and } \( S_2 \) is true

\( S_1 \lor S_2 \) is true iff \( S_1 \) is true \text{ or } \( S_2 \) is true

\( S_1 \Rightarrow S_2 \) is true iff \( S_1 \) is false \text{ or } \( S_2 \) is true

\( i.e., \) \( S_1 \Leftrightarrow S_2 \) is false iff \( S_1 \) is true \text{ and } \( S_2 \) is false

\( S_1 \Leftrightarrow S_2 \) is true iff
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \) \( P_{2,2} \) \( P_{3,1} \)

\[
\begin{array}{ccc}
\text{false} & \text{true} & \text{false} \\
\end{array}
\]

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

\[
\begin{align*}
\neg S & \quad \text{is true iff } S \text{ is false} \\
S_1 \land S_2 & \quad \text{is true iff } S_1 \text{ is true and } S_2 \text{ is true} \\
S_1 \lor S_2 & \quad \text{is true iff } S_1 \text{ is true or } S_2 \text{ is true} \\
S_1 \rightarrow S_2 & \quad \text{is true iff } S_1 \text{ is false or } S_2 \text{ is true} \\
i.e., & \quad \text{is false iff } S_1 \text{ is true and } S_2 \text{ is false} \\
S_1 \leftrightarrow S_2 & \quad \text{is true iff } S_1 \rightarrow S_2 \text{ is true and } S_2 \rightarrow S_1 \text{ is true}
\end{align*}
\]
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model $m$:

$\neg S$ is true iff $S$ is false

$S_1 \land S_2$ is true iff $S_1$ is true and $S_2$ is true

$S_1 \lor S_2$ 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, e.g.,

$\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (true \lor false) = true \land true = true$
### Truth Tables for Connectives

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Wumpus World Sentences

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

- $\neg P_{1,1}$
- $\neg B_{1,1}$
- $B_{2,1}$

- "Pits cause breezes in adjacent squares"

$B_{1,1} \iff (P_{1,2} \lor P_{2,1})$

$B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
Truth Tables for Inference

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$KB$</th>
<th>$\alpha_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Inference by Enumeration

function `TT-ENTAILS?(KB, \alpha)` returns `true` or `false`

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

function `TT-CHECK-ALL(KB, \alpha, symbols, model)` returns `true` or `false`

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

- For `n` symbols, time complexity is $O(2^n)$, space complexity is $O(n)$
Inference by Enumeration

- Sound?
Inference by Enumeration

- Sound? Yes
Inference by Enumeration

- Sound? Yes
  - Entailment is used directly!
Inference by Enumeration

• Sound? Yes
  – Entailment is used directly!
• Complete?
Inference by Enumeration

• Sound? Yes
  – Entailment is used directly!
• Complete? Yes
Inference by Enumeration

• Sound? Yes
  – Entailment is used directly!

• Complete? Yes
  – Works for all KB and a
  – Always stops
Outline

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic
- **Equivalence, validity, satisfiability**
- Inference rules and theorem proving
  - Forward chaining
  - Backward chaining
  - Resolution
Logical Equivalence

- Two sentences are **logically equivalent** iff true in same models:
- \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) and \( \beta \models \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv ((\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv ((\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\lnot(\lnot\alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \rightarrow \beta) & \equiv (\lnot\beta \rightarrow \lnot\alpha) \quad \text{contraposition} \\
(\alpha \rightarrow \beta) & \equiv (\lnot\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \leftrightarrow \beta) & \equiv ((\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha)) \quad \text{biconditional elimination} \\
\lnot(\alpha \land \beta) & \equiv (\lnot\alpha \lor \lnot\beta) \quad \text{de Morgan} \\
\lnot(\alpha \lor \beta) & \equiv (\lnot\alpha \land \lnot\beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Validity and Satisfiability

• A sentence is **valid** if it is true in all models,
  – e.g., *True*, *A ∨ ¬A*, *A ⊃ A*, *(A ∧ (A ⊃ B)) ⊃ B*
  – a.k.a. Tautologies

• Validity is connected to inference via the **Deduction Theorem**:
  – *KB ⊨ α* if and only if (*KB ⊃ α*) is valid
  – Every valid implication sentence describes a legitimate inference

• A sentence is **satisfiable** if it is true in some model
  – e.g., *A ∨ B*

• A sentence is **unsatisfiable** if it is true in no models
  – e.g., *A ∧ ¬A*

• Satisfiability is connected to inference via the following:
  – *KB ⊨ α* if and only if (*KB ∧ ¬α*) is unsatisfiable
  – Proof by contradiction
  – Assume α is false and show this causes a contradiction in KB
Proof Methods

- Proof methods divide into (roughly) two kinds:
  - **Natural Deduction: 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 algorithm
    * Typically require transformation of sentences into a normal form
  - **Model checking**
    * truth table enumeration (always exponential in $n$)
    * improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    * heuristic search in model space (sound but incomplete)
      e.g., min-conflicts like hill-climbing algorithms
Resolution

Conjunctive Normal Form (CNF)
conjunction of disjunctions of literals
clauses
E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- Resolution inference rule (for CNF):

\[
\frac{\ell_i \lor \ldots \lor \ell_k, \quad m_1 \lor \ldots \lor m_n}{\ell_i \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n}
\]

where \(\ell_i\) and \(m_j\) are complementary literals.
E.g., \(P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}\)

\[
P_{1,3}
\]
Resolution

Conjunctive Normal Form (CNF)
conjunction of disjunctions of literals
clauses
E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- **Resolution** inference rule (for CNF):

\[
\begin{array}{c}
\ell_1 \lor \cdots \lor \ell_k, \\
\ell_i \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\end{array}
\]

where \(\ell_i\) and \(m_j\) are complementary literals.
E.g., \(P_{1,3} \lor P_{2,2}, \neg P_{2,2}\)

- Resolution is sound and complete for propositional logic
Resolution

Soundness of resolution inference rule:

\[ \neg (\ell_1 \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k) \Rightarrow \ell_i \]

\[ \neg m_j \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]

\[ \neg (\ell_1 \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k) \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]
Resolution

- Assume \( \ell_i \) is true
  - Then \( m_j \) is false
  - We were given \( m_1 \lor \ldots \lor m_n \)
  - Thus \( m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n \) is true

- Assume \( \ell_i \) is false
  - Then \( m_j \) is true
  - We were given \( \ell_i \lor \ldots \lor \ell_k \)
  - Thus \( \ell_i \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k \) is true
Conversion to CNF

• Resolution rule can derive any conclusion entailed by any propositional knowledge base
• But only works for disjunctions of literals!
• We want to work with KBs that have statements in other forms
• What to do?
Conversion to CNF

- Fortunately, every propositional logic sentence is equivalent to a conjunction of disjunctive literals
  - Known as Conjunctive Normal Form (CNF)
Conversion to CNF

$B_{1,1} \iff (P_{1,2} \lor P_{2,1})$

1. Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \implies \beta) \land (\beta \implies \alpha)$.
   
   $(B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1})$

2. Eliminate $\implies$, replacing $\alpha \implies \beta$ with $\neg \alpha \lor \beta$.
   
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:
   
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributivity law ($\land$ over $\lor$) and flatten:
   
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$
Resolution Algorithm

- Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable
  - Recall $KB \models \alpha$ if and only if $(KB \land \neg \alpha)$ is unsatisfiable

```plaintext
function PL-Resolution(KB, \alpha) returns true or false
    clauses ← the set of clauses in the CNF representation of $KB \land \neg \alpha$
    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
```
Resolution Algorithm

• Why is empty clause equivalent to False?
Resolution Algorithm

- Why is empty clause equivalent to False?
- Disjunction only true if one value is false
- Also, only happens if include P and not P
Resolution Example

- $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \land \alpha = \neg P_{1,2}$
Outline

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Forward chaining
  - Backward chaining
  - Resolution
Forward and Backward Chaining

- Often don’t need full power of resolution because data is in Horn Form
  - KB = conjunction of Horn clauses
  - Horn clause =
    - Disjunction with **at most one** positive literal, or
    - proposition symbol, or
    - (conjunction of symbols) ⇒ symbol
  - E.g., C ∧ (B ⇒ A) ∧ (C ∧ D ⇒ B)

- **Modus Ponens** (for Horn Form): complete for Horn KBs
  \[
  \alpha_1, \ldots, \alpha_n, \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta
  \]

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

- Idea: fire any rule whose premises are satisfied in the $KB$,
  - Add its conclusion to the $KB$, until query is found

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A \\
B
\end{align*}
\]
Forward Chaining Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
    local variables: count, a table, indexed by clause, initially the number of premises
    inferred, a table, indexed by symbol, each entry initially false
    agenda, a list of symbols, initially the symbols known to be true

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

- Forward chaining is sound and complete for Horn KB
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
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 $m$
  4. Assume some $a_1 \land \ldots \land a_k \Rightarrow b$ is false
     Then $a_1 \land \ldots \land a_k$ is true and is $b$ false; contradicting 1)
  5. Hence $m$ is a model of $KB$
  6. If $KB \models q$, $q$ is true in every model of $KB$, including $m$
Outline

• Knowledge-based agents
• Wumpus world
• Logic in general - models and entailment
• Propositional (Boolean) logic
• Equivalence, validity, satisfiability
• Inference rules and theorem proving
  – Forward chaining
  – **Backward chaining**
  – Resolution
Backward Chaining

- Idea: 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
  - has already been proved true, or
  - has already failed
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example

Diagram showing nodes and connections.
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Forward vs. Backward Chaining

• FC is **data-driven**, 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

• Generally, agents share work between both
  – Limit forward reasoning to find facts that are relevant while backwards chaining
Efficient Propositional Inference

- Two families of efficient algorithms for propositional inference:
  - Complete backtracking search algorithms
    - DPLL algorithm (Davis, Putnam, Logemann, Loveland)
  - Incomplete local search algorithms
    - WalkSAT algorithm
The DPLL Algorithm

• Determine if an input propositional logic sentence (in CNF) is satisfiable.

• Improvements over truth table enumeration:
  1. Early termination
     • A clause is true if any literal is true.
     • A sentence is false if any clause is false.
  2. Pure symbol heuristic
     • Pure symbol: always appears with the same "sign" in all clauses.
     • e.g., In the three clauses \((A \lor \neg B), (\neg B \lor \neg C), (C \lor A)\), \(A\) and \(B\) are pure, \(C\) is impure.
     • Make a pure symbol literal true.
  3. Unit clause heuristic
     • Unit clause: only one literal in the clause
     • The only literal in a unit clause must be true.
The DPLL Algorithm

function DPLL-Satisfiable?(s) returns true or false
    inputs: s, a sentence in propositional logic
    clauses $\leftarrow$ the set of clauses in the CNF representation of s
    symbols $\leftarrow$ a list of the proposition symbols in s
    return DPLL(clauses, symbols, [])

function DPLL(clauses, symbols, model) returns true or false
    if every clause in clauses is true in model then return true
    if some clause in clauses is false in model then return false
    $P, value \leftarrow$ FIND-PURE-SYMBOL(symbols, clauses, model)
    if $P$ is non-null then return DPLL(clauses, symbols-$P$, $[P = value | model]$)
    $P, value \leftarrow$ FIND-UNIT-CLAUSE(clauses, model)
    if $P$ is non-null then return DPLL(clauses, symbols-$P$, $[P = value | model]$)
    $P \leftarrow$ FIRST(symbols); rest $\leftarrow$ REST(symbols)
    return DPLL(clauses, rest, $[P = true | model]$) or
        DPLL(clauses, rest, $[P = false | model]$)
The *WalkSAT* Algorithm

- Incomplete, local search algorithm
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- Balance between greediness and randomness
The \textbf{WalkSAT} Algorithm

\begin{verbatim}
function \textsc{WalkSAT}(\textit{clauses}, p, \textit{max-flips}) returns a satisfying model or \textit{failure}
inputs: \textit{clauses}, a set of clauses in propositional logic
        \(p\), the probability of choosing to do a “random walk” move
        \textit{max-flips}, number of flips allowed before giving up

\textit{model} \leftarrow \text{a random assignment of true/false to the symbols in } \textit{clauses}
for \(i = 1\) to \textit{max-flips} do
    if \textit{model} satisfies \textit{clauses} then return \textit{model}
    \textit{clause} \leftarrow \text{a randomly selected clause from } \textit{clauses} \text{ that is false in } \textit{model}
    with probability \(p\) flip the value in \textit{model} of a randomly selected symbol from \textit{clause}
else flip whichever symbol in \textit{clause} maximizes the number of satisfied clauses
return \textit{failure}
\end{verbatim}
Hard Satisfiability Problems

• Consider random 3-CNF sentences. e.g.,

\[(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)\]

\[m = \text{number of clauses}\]
\[n = \text{number of symbols}\]

– Hard problems seem to cluster near \(m/n = 4.3\) (critical point)
Hard Satisfiability Problems

![Graph showing the probability of satisfiability vs the clause/symbol ratio m/n.](image)
Hard Satisfiability Problems

- Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$
Inference-based Agents in The Wumpus World

• A wumpus-world agent using propositional logic:

\[
\begin{align*}
\neg P_{1,1} \\
\neg W_{1,1} \\
B_{x,y} &\iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \\
S_{x,y} &\iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \\
W_{1,1} &\lor W_{1,2} \lor \ldots \lor W_{4,4} \\
\neg W_{1,1} &\lor \neg W_{1,2} \\
\neg W_{1,1} &\lor \neg W_{1,3} \\
\ldots
\end{align*}
\]

• On a 4x4 board:
  – 64 distinct proposition symbols
  – 155 sentences
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} \land \neg W_{i,j})) is true or
        for some fringe square [i,j], ASK(KB, (P_{i,j} \lor 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
Expressiveness Limitation of Propositional Logic

• KB contains "physics" sentences for every single square

• For every time $t$ and every location $[x,y]$, 
  \[ L_{x,y} \land \text{FacingRight}^t \land \text{Forward}^t \Rightarrow L_{x+1,y}^t \]

• Rapid proliferation of clauses
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 w.r.t. models
  - Entailment: necessary truth of one sentence given another
  - Inference: deriving sentences from other sentences
  - Soundness: derivations produce only entailed sentences
  - Completeness: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Resolution is complete for propositional logic
  - Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power to scale well