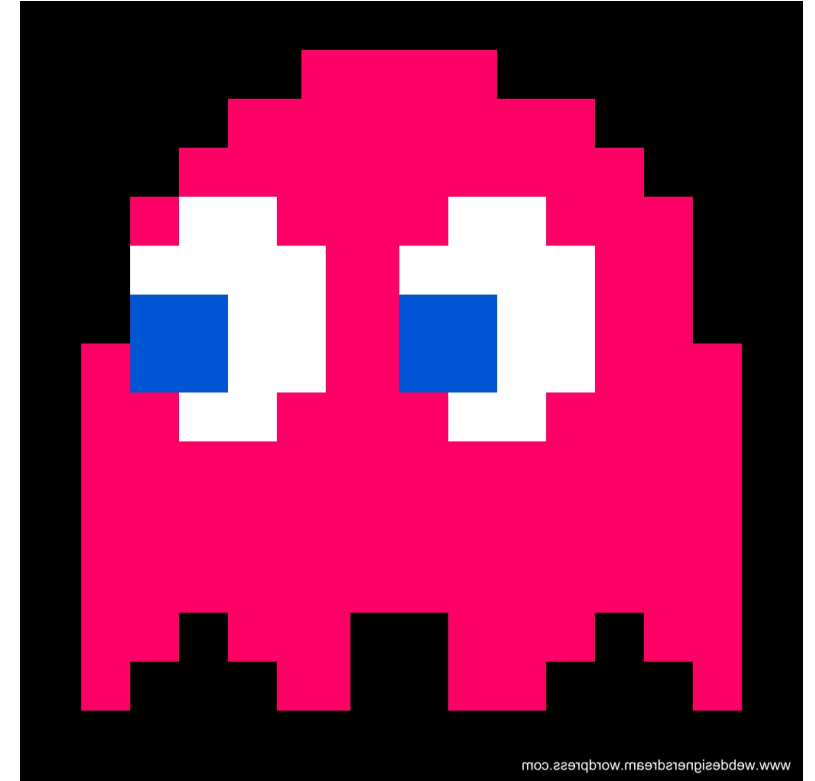
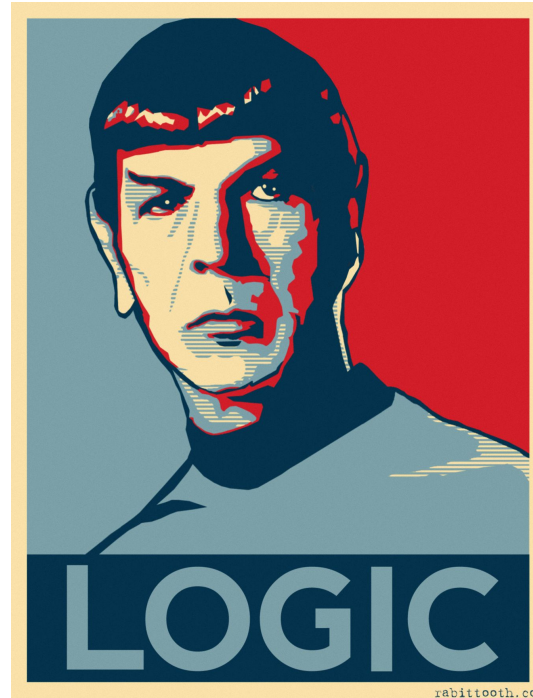


CS 4700: Foundations of Artificial Intelligence

Propositional Logic I

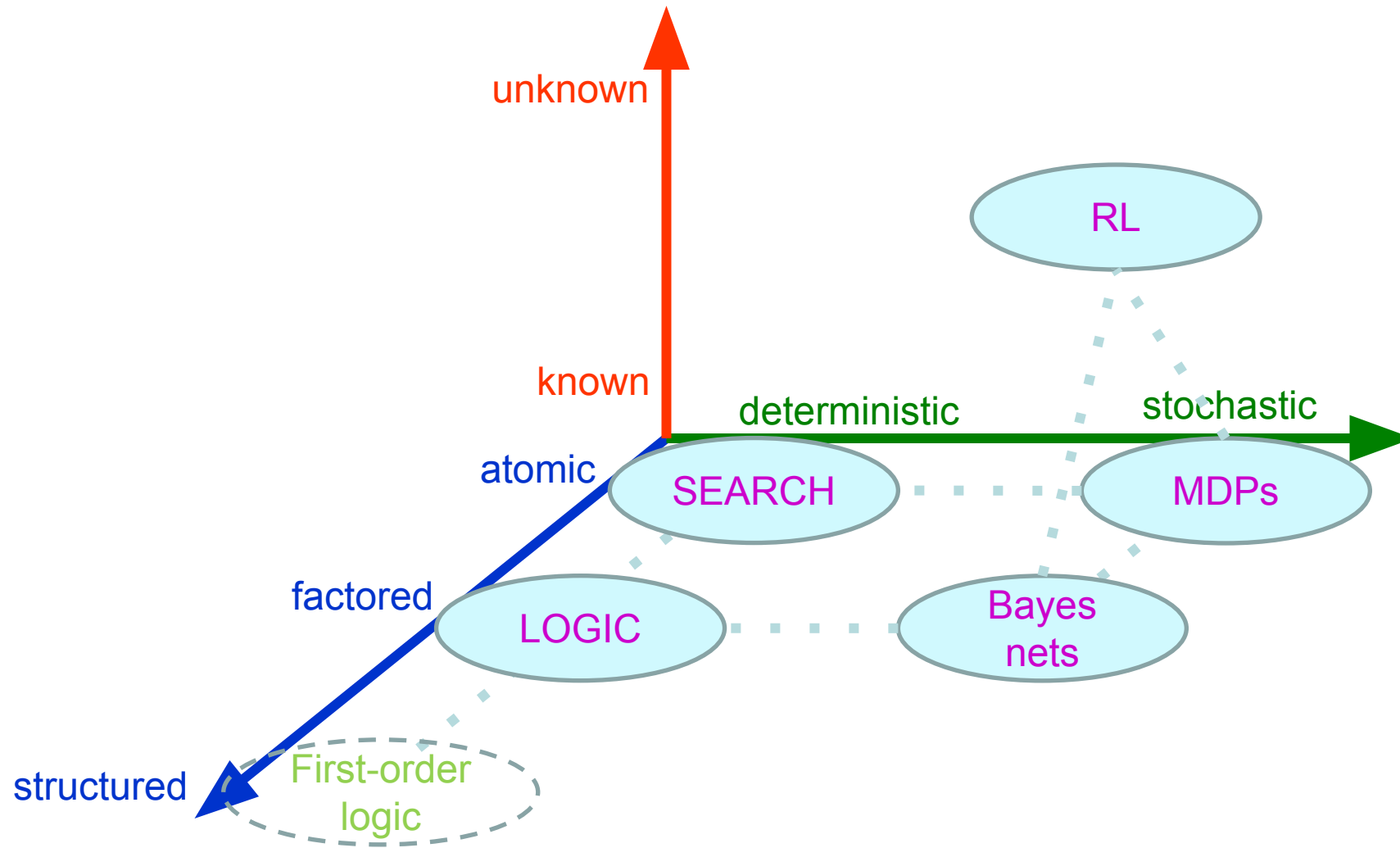


Instructors: Kevin Ellis

Cornell University

Slides adapted from Berkeley CS188: ai.berkeley.edu

Outline of the course (minus learning)



Problems with neural networks

Cannot modularly add new knowledge

Cannot inspect and debug inner workings

Cannot reliably extrapolate and generalize beyond training set

Can produce contradictory answers

Require vast amounts of data

Outline

1. Propositional Logic I

- Basic concepts of knowledge, logic, reasoning
- Propositional logic: syntax and semantics, Pacworld example
- Inference by theorem proving

2. Propositional logic II

- Inference by model checking
- A Pac agent using propositional logic

3. First-order logic

Agents that know things

- Agents acquire knowledge through perception, learning, language
 - Knowledge of the effects of actions (“transition model”)
 - Knowledge of how the world affects sensors (“sensor model”)
 - Knowledge of the current state of the world
- Can keep track of a partially observable world
- Can formulate plans to achieve goals
- Can design and build gravitational wave detectors.....

Knowledge, contd.

- 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 (or have it **Learn** the knowledge)
 - 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
- A single inference algorithm can answer any answerable question

Knowledge base

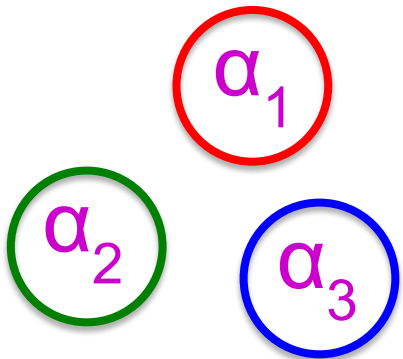
Inference engine

Domain-specific facts

Generic code

Logic

- **Syntax**: What sentences are allowed?
- **Semantics**:
 - What are the **possible worlds**?
 - Which sentences are **true** in which worlds? (i.e., **definition** of truth)



Syntaxland



Semanticsland

Different kinds of logic

- Propositional logic

- Syntax: $P \vee (\neg Q \wedge R)$; $X_1 \Leftrightarrow (\text{Raining} \Rightarrow \neg \text{Sunny})$
- Possible world: $\{P=\text{true}, Q=\text{true}, R=\text{false}, S=\text{true}\}$ or 1101
- Semantics: $\alpha \wedge \beta$ is true in a world iff α is true and β is true (etc.)

- First-order logic

- Syntax: $\forall x \exists y P(x,y) \wedge \neg Q(\text{Joe}, f(x)) \Rightarrow f(x)=f(y)$
- Possible world: Objects o_1, o_2, o_3 ; P holds for $\langle o_1, o_2 \rangle$; Q holds for $\langle o_3 \rangle$; $f(o_1)=o_1$; $\text{Joe}=o_3$; etc.
- Semantics: $\varphi(\sigma)$ is true in a world if $\sigma=o_j$ and φ holds for o_j ; etc.

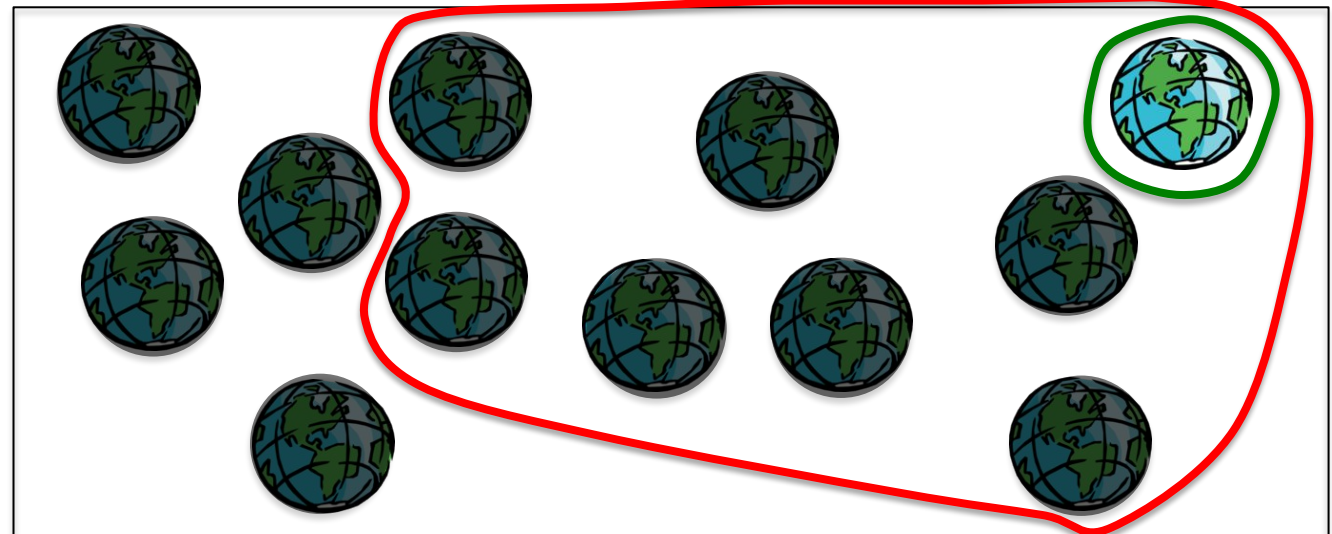
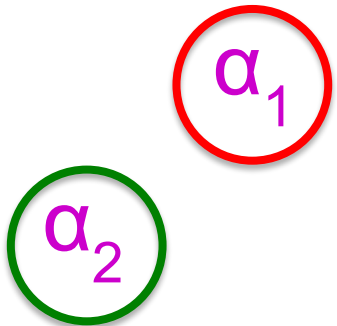
Different kinds of logic, contd.

- Relational databases:

- Syntax: ground relational sentences, e.g., *Sibling(Ali,Bo)*
- Possible worlds: (typed) objects and (typed) relations
- Semantics: sentences in the DB are true, everything else is false
 - Cannot express disjunction, implication, universals, etc.
 - Query language (SQL etc.) typically some variant of first-order logic
 - Often augmented by first-order rule languages, e.g., Datalog
- Knowledge graphs (roughly: relational DB + ontology of types and relations)
 - Google Knowledge Graph: 5 billion entities, 500 billion facts
 - Facebook network: 2.8 billion people, trillions of posts, maybe quadrillions of facts

Inference: entailment

- **Entailment:** $\alpha \models \beta$ (“ α entails β ” or “ β follows from α ”) iff in every world where α is true, β is also true
 - I.e., the α -worlds are a subset of the β -worlds [$\text{models}(\alpha) \subseteq \text{models}(\beta)$]
- In the example, $\alpha_2 \models \alpha_1$
- (Say α_2 is $\neg Q \wedge R \wedge S \wedge W$
 α_1 is $\neg Q$)



Inference: proofs

- A proof is a **demonstration** of entailment between α and β
- **Sound** algorithm: everything it claims to prove is in fact entailed
- **Complete** algorithm: every that is entailed can be proved

Inference: proofs

- Method 1: *model-checking*

- For every possible world, if α is true make sure that β is true too
- OK for propositional logic (finitely many worlds); not easy for first-order logic

- Method 2: *theorem-proving*

- Search for a sequence of proof steps (applications of *inference rules*) leading from α to β
- E.g., from $P \wedge (P \Rightarrow Q)$, infer Q by *Modus Ponens*

Propositional logic syntax

- Given: a set of proposition symbols $\{X_1, X_2, \dots, X_n\}$
 - (we often add **True** and **False** for convenience)
- X_i is a sentence
- If α is a sentence then $\neg\alpha$ is a sentence
- If α and β are sentences then $\alpha \wedge \beta$ is a sentence
- If α and β are sentences then $\alpha \vee \beta$ is a sentence
- If α and β are sentences then $\alpha \Rightarrow \beta$ is a sentence
- If α and β are sentences then $\alpha \Leftrightarrow \beta$ is a sentence
- And p.s. there are no other sentences!

Propositional logic semantics

- Let m be a model assigning true or false to $\{X_1, X_2, \dots, X_n\}$
- If α is a symbol then its truth value is given in m
- $\neg\alpha$ is true in m iff α is false in m
- $\alpha \wedge \beta$ is true in m iff α is true in m and β is true in m
- $\alpha \vee \beta$ is true in m iff α is true in m or β is true in m
- $\alpha \Rightarrow \beta$ is true in m iff α is false in m or β is true in m
- $\alpha \Leftrightarrow \beta$ is true in m iff $\alpha \Rightarrow \beta$ is true in m and $\beta \Rightarrow \alpha$ is true in m

Propositional logic semantics

- $\alpha \Rightarrow \beta$ is true in m iff α is false in m or β is true in m

Person \Rightarrow Mortal

(\neg Person) \vee Mortal

Person \Rightarrow TheSkylsAboveUs

(everything implies something that is definitely true)

OneEqualsTwo \Rightarrow Mortal

(something that's definitely false implies anything)

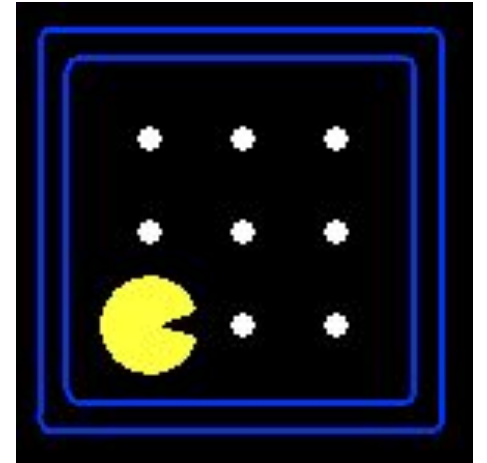
Propositional logic semantics in code

```
function PL-TRUE?( $\alpha$ , model) returns true or false
  if  $\alpha$  is a symbol then return Lookup( $\alpha$ , model)
  if Op( $\alpha$ ) =  $\neg$  then return not(PL-TRUE?(Arg1( $\alpha$ ), model))
  if Op( $\alpha$ ) =  $\wedge$  then return and(PL-TRUE?(Arg1( $\alpha$ ), model),
                                     PL-TRUE?(Arg2( $\alpha$ ), model))
  etc.
```

(Sometimes called “recursion over syntax”)

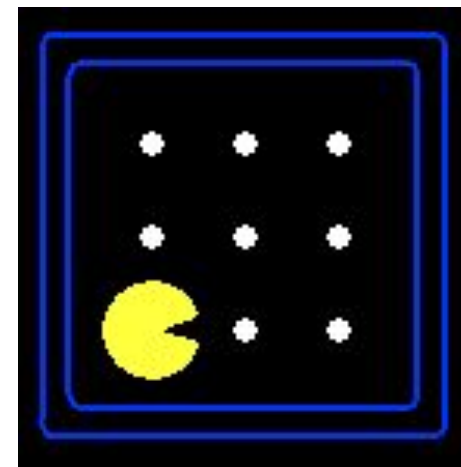
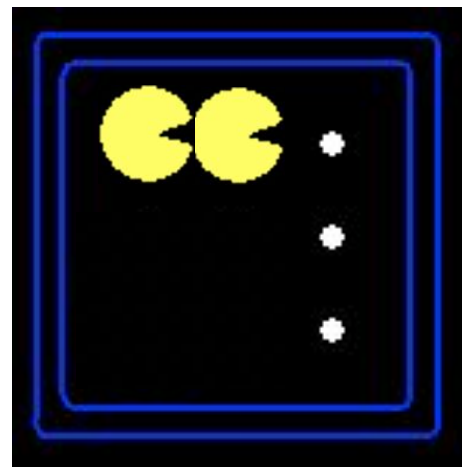
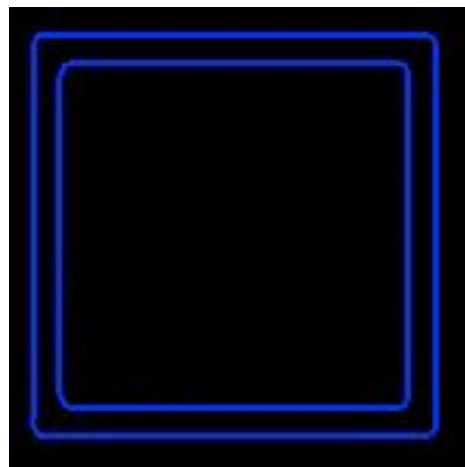
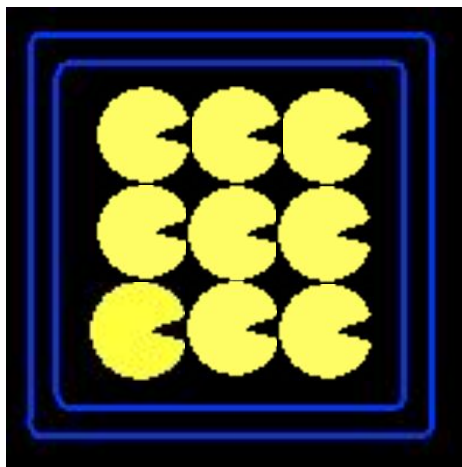
Example: Partially observable Pacman

- Pacman knows the map but perceives just wall/gap to NSEW
- Formulation: *what variables do we need?*
 - Wall locations
 - $Wall_{0,0}$ there is a wall at $[0,0]$
 - $Wall_{0,1}$ there is a wall at $[0,1]$, etc. (N symbols for N locations)
 - Percepts
 - ~~■ $Blocked_W$ (blocked by wall to my West) etc.~~
 - $Blocked_W_0$ (blocked by wall to my West at time 0) etc. ($4T$ symbols for T time steps)
 - Actions
 - W_0 (Pacman moves West at time 0), E_0 etc. ($4T$ symbols)
 - Pacman's location
 - $At_{0,0}_0$ (Pacman is at $[0,0]$ at time 0), $At_{0,1}_0$ etc. (NT symbols)



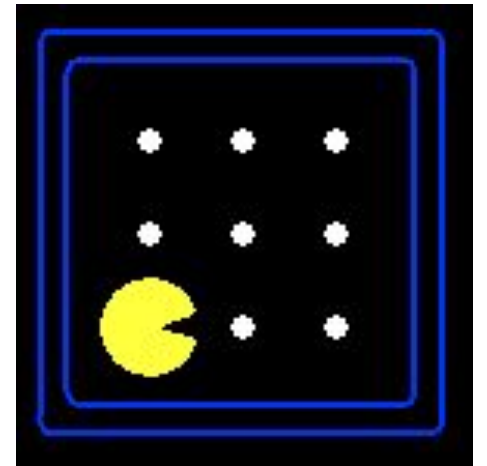
How many possible worlds?

- N locations, T time steps $\Rightarrow N + 4T + 4T + NT = O(NT)$ variables
- $O(2^{NT})$ possible worlds!
- $N=200$, $T=400 \Rightarrow \sim 10^{24000}$ worlds
- Each world is a complete “history”
 - But most of them are pretty weird!



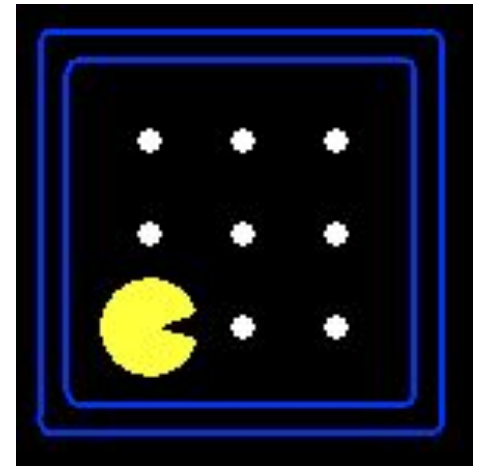
Pacman's knowledge base: Map

- Pacman knows where the walls are:
 - $\text{Wall}_{0,0} \wedge \text{Wall}_{0,1} \wedge \text{Wall}_{0,2} \wedge \text{Wall}_{0,3} \wedge \text{Wall}_{0,4} \wedge \text{Wall}_{1,4} \wedge \dots$
- Pacman knows where the walls aren't!
 - $\neg \text{Wall}_{1,1} \wedge \neg \text{Wall}_{1,2} \wedge \neg \text{Wall}_{1,3} \wedge \neg \text{Wall}_{2,1} \wedge \neg \text{Wall}_{2,2} \wedge \dots$



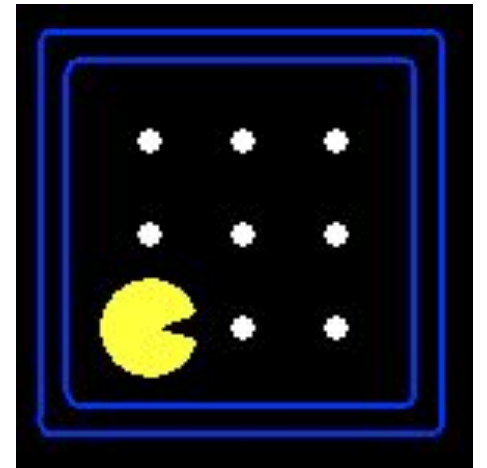
Pacman's knowledge base: Initial state

- Pacman doesn't know where he is
- But he knows he's somewhere!
 - $At_{1,1}_0 \vee At_{1,2}_0 \vee At_{1,3}_0 \vee At_{2,1}_0 \vee \dots$



Pacman's knowledge base: Sensor model

- State facts about how Pacman's percepts arise...
 - $\langle \text{Percept variable at } t \rangle \Leftrightarrow \langle \text{some condition on world at } t \rangle$
- Pacman perceives a wall to the West at time t *if and only if* he is in x,y and there is a wall at $x-1,y$
 - $\text{Blocked_W_0} \Leftrightarrow ((\text{At_1,1_0} \wedge \text{Wall_0,1}) \vee (\text{At_1,2_0} \wedge \text{Wall_0,2}) \vee (\text{At_1,3_0} \wedge \text{Wall_0,3}) \vee \dots)$
 - 4T sentences, each of size $O(N)$
 - Note: these are valid for any map



Pacman's knowledge base: Transition model

- How does each **state variable** at each time gets its value?
 - Here we care about location variables, e.g., $At_{3,3}_{17}$
- A state variable X gets its value according to a **successor-state axiom**
 - $X_t \Leftrightarrow [X_{t-1} \wedge \neg(\text{some action}_{t-1} \text{ made it false})] \vee [\neg X_{t-1} \wedge (\text{some action}_{t-1} \text{ made it true})]$
- For Pacman location:
 - $At_{3,3}_{17} \Leftrightarrow [At_{3,3}_{16} \wedge \neg((\neg Wall_{3,4} \wedge N_{16}) \vee (\neg Wall_{4,3} \wedge E_{16}) \vee \dots)]$
 $\vee [\neg At_{3,3}_{16} \wedge ((At_{3,2}_{16} \wedge \neg Wall_{3,3} \wedge N_{16}) \vee (At_{2,3}_{16} \wedge \neg Wall_{3,3} \wedge N_{16}) \vee \dots)]$

How many sentences?

- Vast majority of KB occupied by $O(NT)$ transition model sentences
 - Each about 10 lines of text
 - $N=200, T=400 \Rightarrow \sim 800,000$ lines of text, or 20,000 pages
- This is because propositional logic has limited expressive power
- Are we really going to write 20,000 pages of logic sentences???
- No, but your code will generate all those sentences!
- In first-order logic, we need $O(1)$ transition model sentences
- (State-space search uses atomic states: how do we keep the transition model representation small???)

A knowledge-based agent

function KB-AGENT(**percept**) **returns** an action

persistent: KB, a knowledge base

t, an integer, initially 0

TELL(KB, MAKE-PERCEPT-SENTENCE(**percept**, t))

action \leftarrow ASK(KB, MAKE-ACTION-QUERY(t))

TELL(KB, MAKE-ACTION-SENTENCE(**action**, t))

t \leftarrow t+1

return **action**

Some reasoning tasks

- **Localization** with a map and local sensing:
 - Given an initial KB, plus a sequence of percepts and actions, where am I?
- **Mapping** with a location sensor:
 - Given an initial KB, plus a sequence of percepts and actions, what is the map?
- **Simultaneous localization and mapping**:
 - Given ..., where am I and what is the map?
- **Planning**:
 - Given ..., what action sequence is guaranteed to reach the goal?
- **ALL OF THESE USE THE SAME KB AND THE SAME ALGORITHM!!**

Summary

- One possible agent architecture: knowledge + inference
- Logics provide a formal way to encode knowledge
 - A logic is defined by: syntax, set of possible worlds, truth condition
- A simple KB for Pacman covers the initial state, sensor model, and transition model
- Logical inference computes entailment relations among sentences, enabling a wide range of tasks to be solved