CSC242: Introduction to Artificial Intelligence

Lecture 2.2

Please put away all electronic devices

Constraint Satisfaction Problem (CSP)

- X: Set of variables $\{ X_1, ..., X_n \}$
- D: Set of domains $\{ D_1, ..., D_n \}$
  - Each D_i : set of values $\{ v_1, ..., v_k \}$
- C: Set of constraints $\{ C_1, ..., C_m \}$
- Solution: Assign to each X_i a value from D_i such that all the C_i are satisfied

Factored Representation

- Splits a state into variables (or attributes) that can have values
- Factored states can be more or less similar (unlike atomic states)
- Can also represent uncertainty (don't know value of some attribute)

Backtracking Search

- DFS search through the space of assignments
- Assign one variable at a time
- Because the representation of CSPs is standardized, no need to supply initial state, actions, transition model, or goal test!

Constraint Propagation

- Using the constraints to reduce the set of legal values of a variable, which can in turn reduce the legal values of another variable, and so on
- Not a search process!
- Part of state update in state-space search
- A type of inference: making implicit information explicit

Node Consistency

- Apply all unary constraints
- If problem is not inconsistent, then we can always propagate unary constraints at the start
- And then we can ignore them
- Complexity: Each variable, each value, each unary constraint

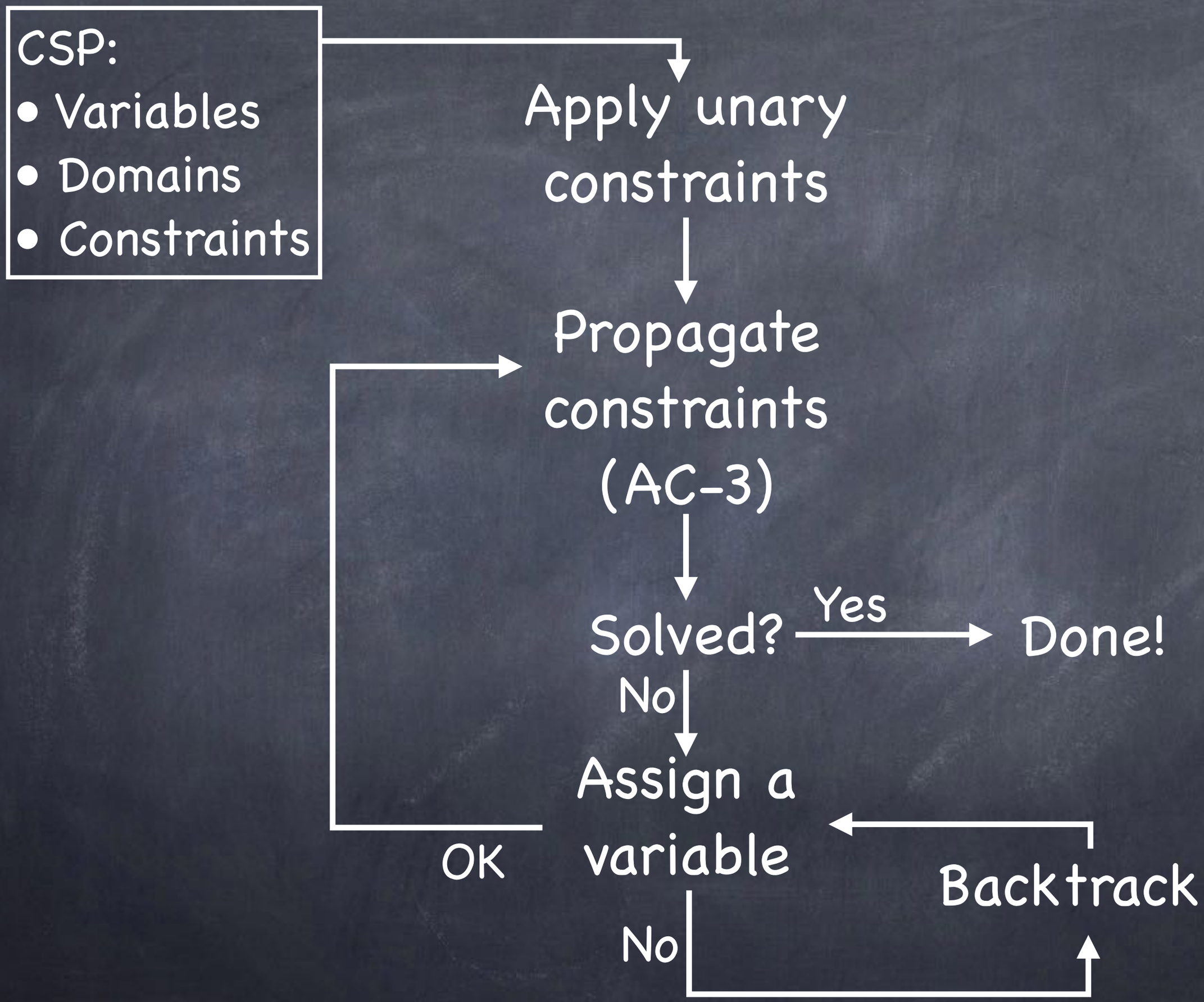
Arc Consistency

X_i is arc-consistent w.r.t. X_j if

for <u>every</u> value in the domain D_i,

there is <u>some</u> value in the domain D_j

that satisfies the binary constraint on the arc (X_i, X_j)


CSP:
• Variables
• Domains
• Constraints
Apply unary constraints
Propagate constraints (AC-3)
Solved?
Yes
Done!
No
Assign a variable
OK
Backtrack
No

CSP Secret Sauce

- Factored representation of state:
  - Variables, Domains, Constraints
- Allows:
  - Early pruning of inconsistent states
  - Inference during search to reduce alternatives

Hunt The Wumpus

```
Hunt the Wumpus
Original BASIC version (1972) by Gregory Yob.
Inform port (1999) by Magnus Olsson <zebulon@pobox.com>.
Release 1 / Serial number 991216 / Inform v6.21

Type 1 to read the instructions, 2 to read the implementation n
the game, or 4 to quit.
3

You can choose between the following caves:
1: The Dodecahedron
2: The Möbius Strip
3: The String of Beads
4: The Dendrite
5: The One-way Lattice
Which cave (1-5)? 1
OK, using the Dodecahedron

Bats nearby!
You are in room 1
Tunnels lead to 2 5 8
Shoot, Move or Quit (S-M-Q)? _
```

C:\Documents and Settings\jatwood\Desktop\Wumpus_vsnet_solution\HuntTheWump...

```
*** Wumpus .NET ***

Based on:


                        WUMPUS 2
                    CREATIVE COMPUTING
                  MORRISTOWN  NEW JERSEY



INSTRUCTIONSn
CAVE #(0-6) 1

HUNT THE WUMPUS

YOU ARE IN ROOM 13
TUNNELS LEAD TO 11 14 15

SHOOT OR MOVE m
WHERE TO 14


I FEEL A DRAFT!
YOU ARE IN ROOM 14
TUNNELS LEAD TO 12 13 16

SHOOT OR MOVE m
WHERE TO 13


YOU ARE IN ROOM 13
TUNNELS LEAD TO 11 14 15

SHOOT OR MOVE m
WHERE TO 11


I SMELL A WUMPUS!
YOU ARE IN ROOM 11
TUNNELS LEAD TO 12 13 9

SHOOT OR MOVE s
NO. OF ROOMS 1
ROOM #12

MISSED

YOU ARE IN ROOM 11
TUNNELS LEAD TO 12 13 9
```

TEXAS INSTRUMENTS

HOME COMPUTER

HUNT THE WUMPUS

ARCADE ENTERTAINMENT

SOLID STATE CARTRIDGE

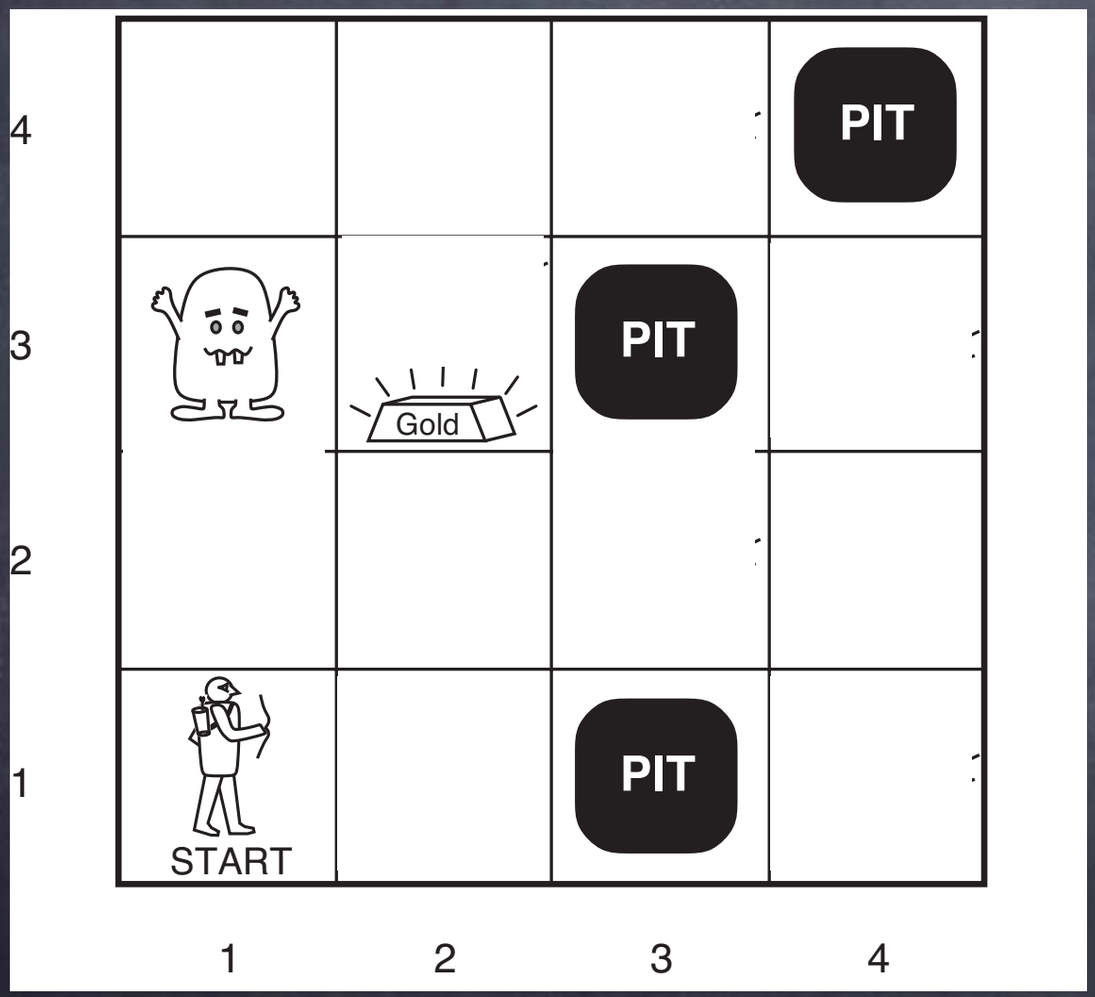
This game can be played using the optional Wired Remote Controllers.

An exciting simulated hunt in a hidden maze of caverns and twisting tunnels. Seek out the lair of the Wumpus, while avoiding perils along the way!


Hunt the Wumpus

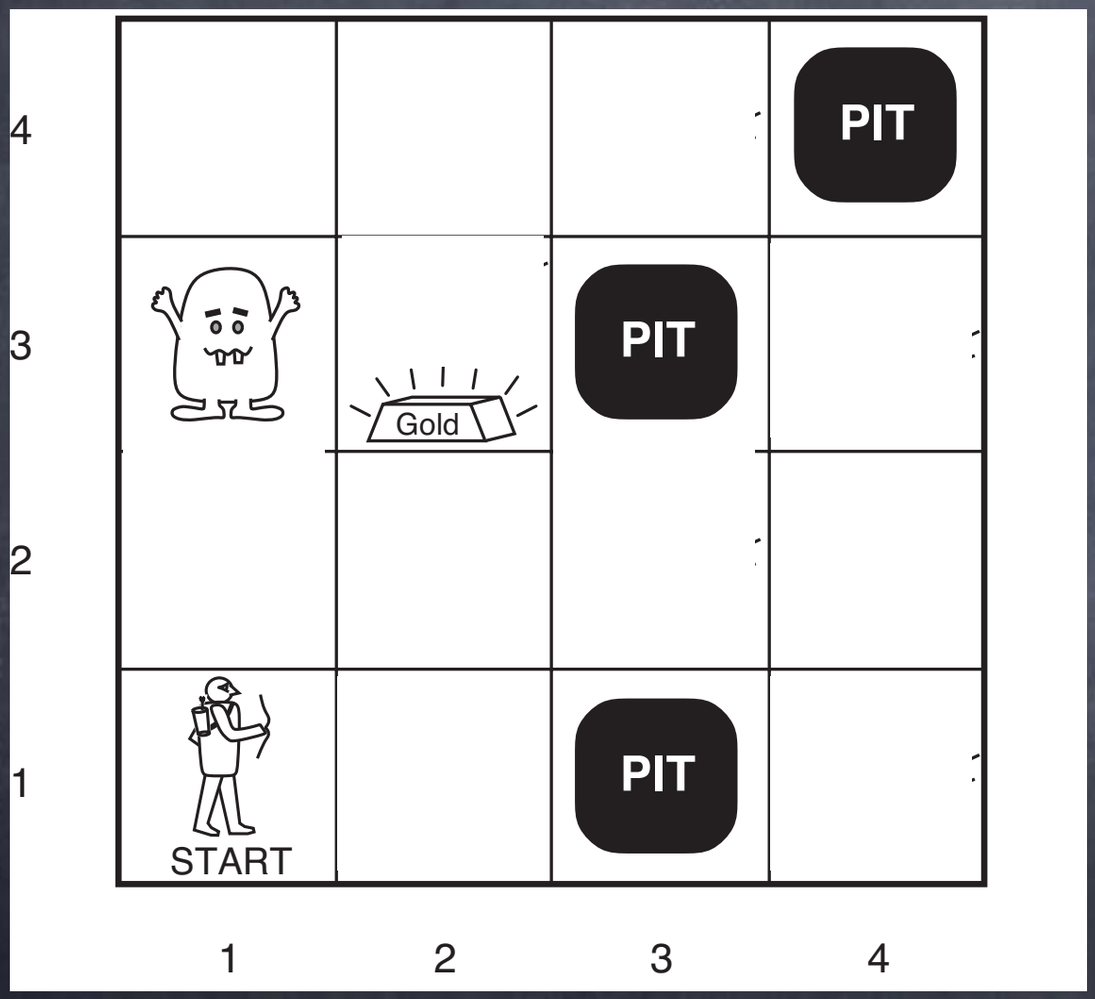

Hunted Wumpus
3
Creature — Beast
X
When Hunted Wumpus comes into play, each other player may put a creature card from his or her hand into play.
Just one can feed a dozen people for a month.
6/6
Thomas M. Baxa

Hunt The Wumpus



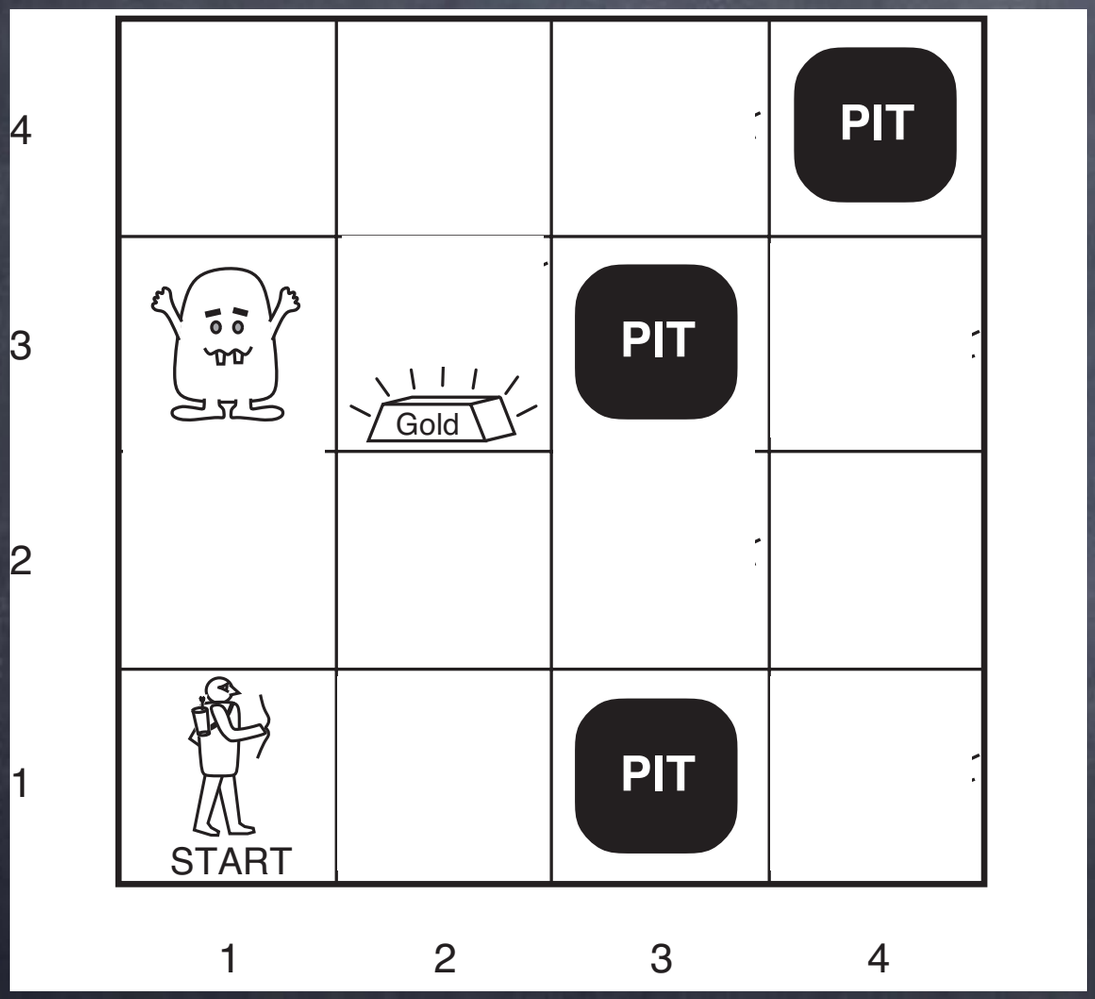
- $n \times n$ grid of rooms
- Agent starts in [1,1] facing left
- Gold, wumpus in random square (not [1,1])
- Wumpus doesn't move
- Pits random ($P=0.2$)

Hunt The Wumpus



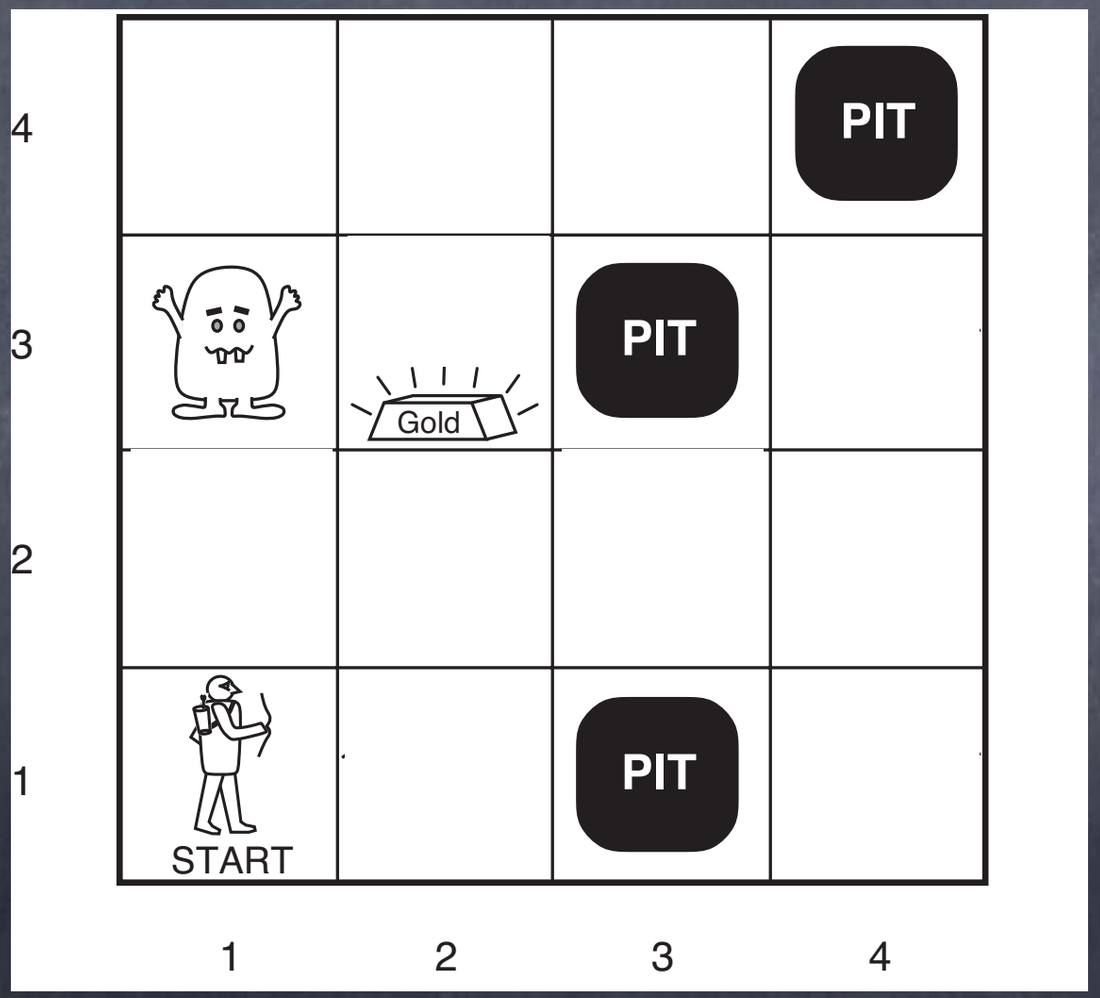
- Move fwd, turn left/ right
- Grab gold
- Shoot arrow (once)
- Climb out (from [1,1])
- Costs:
  - -1 per move
  - -10 to shoot

Hunt The Wumpus

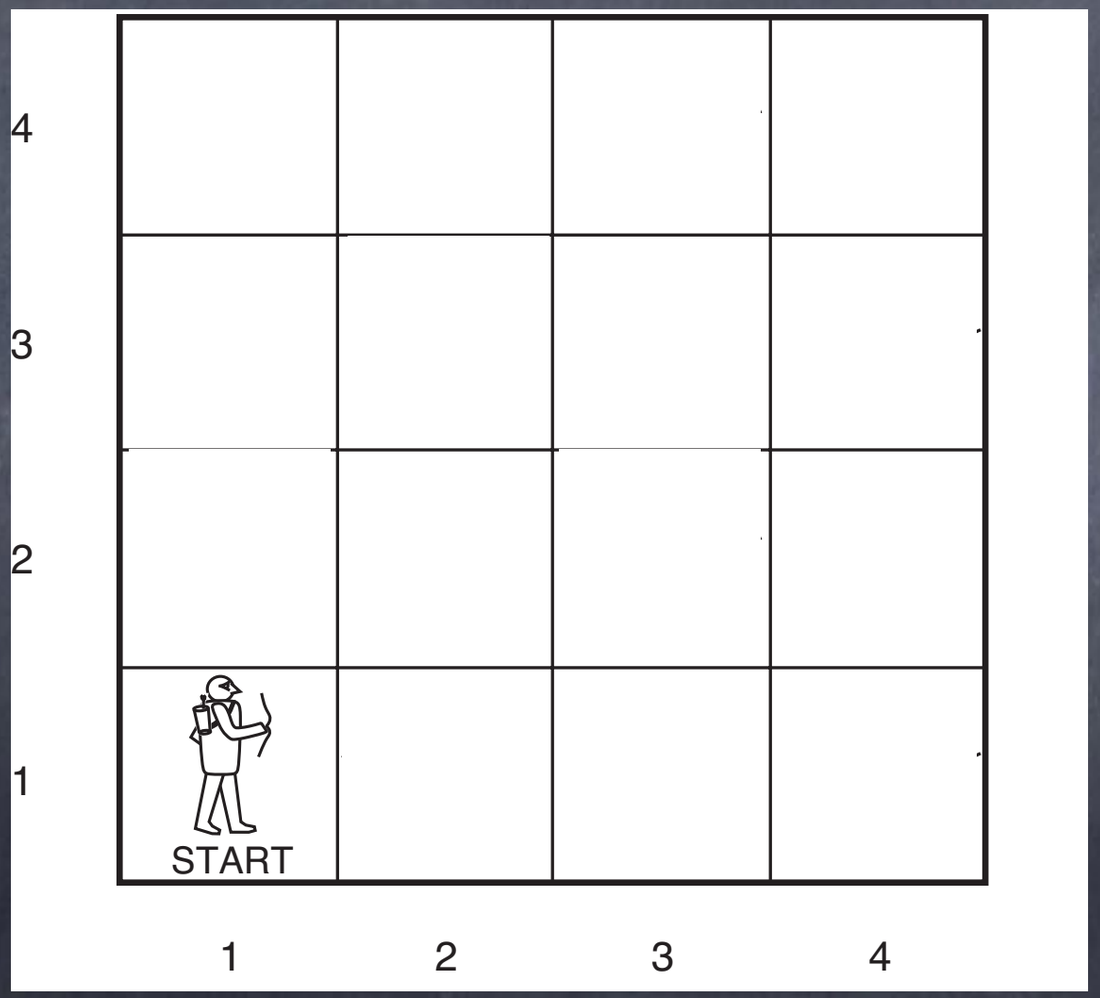


- Get gold and climb out: +1000
- Fall in pit or get eaten by wumpus: -1000

Hunt The Wumpus

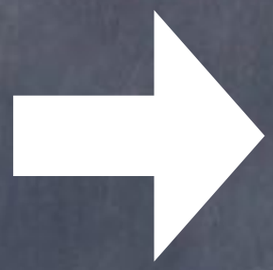


Hunt The Wumpus



Partially Observable

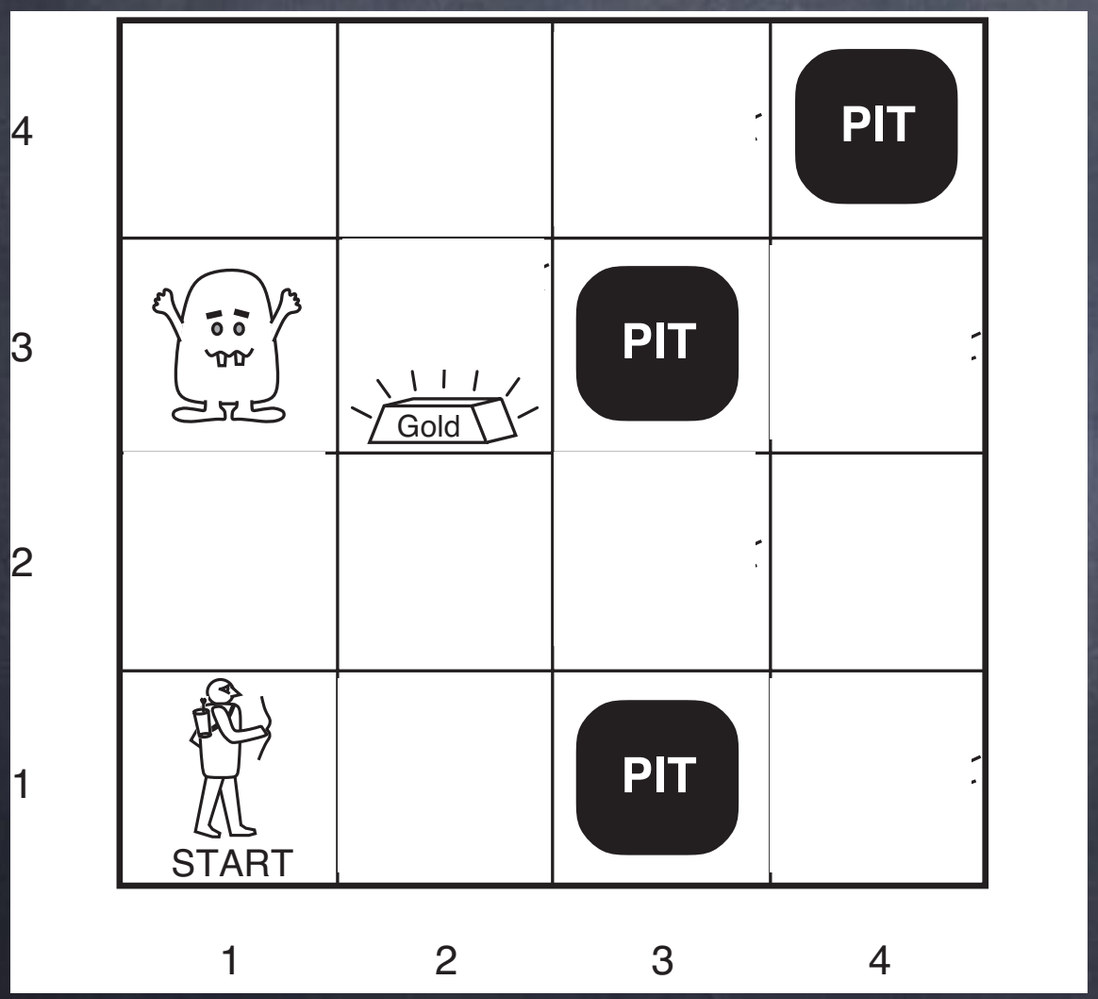
Don't know everything about the state of the world



Have to represent uncertainty in our knowledge

Have to explore

Representation



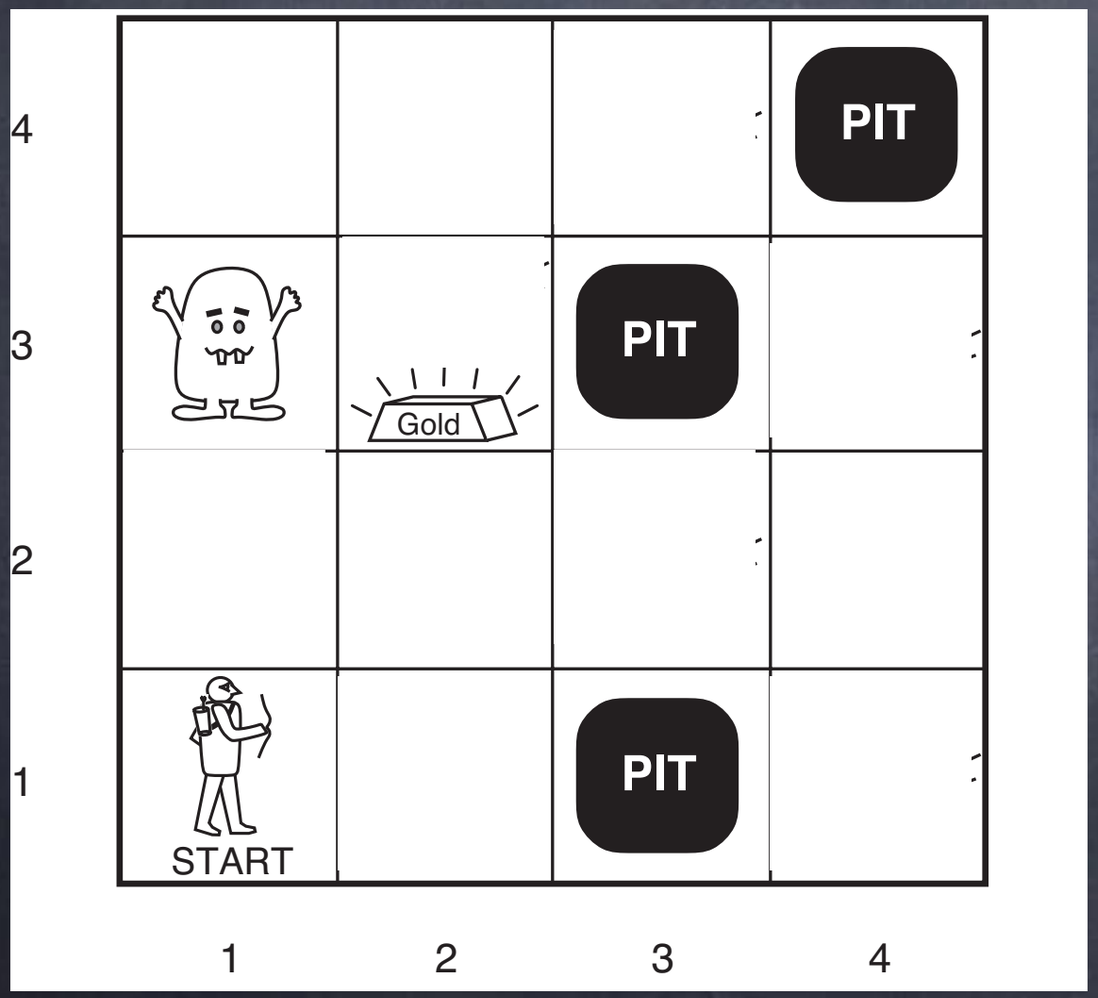
For each room:

Has pit? true or false

Has gold? true or false

Has wumpus? true or false

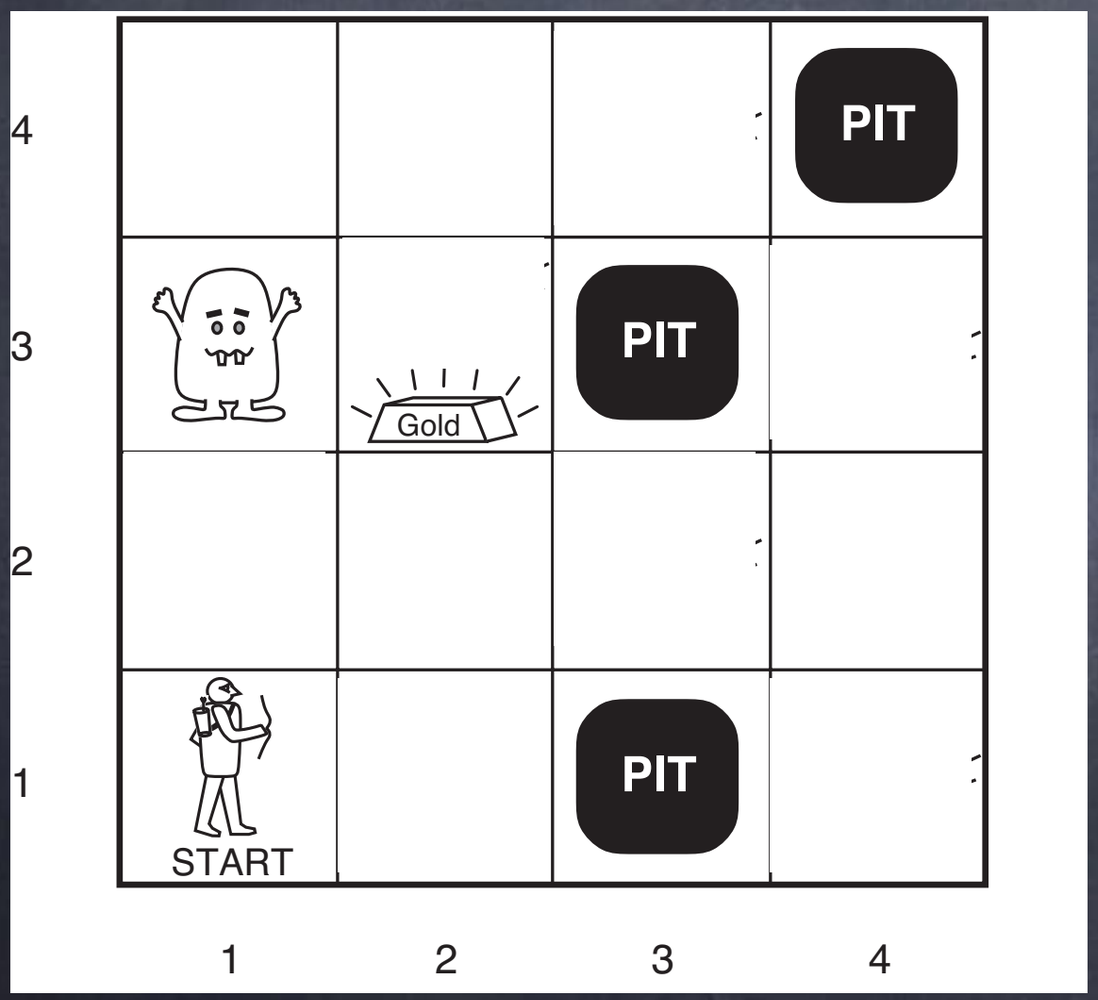
Representation




```
Boolean[][][] env =
     new Boolean[n][n][3];
```

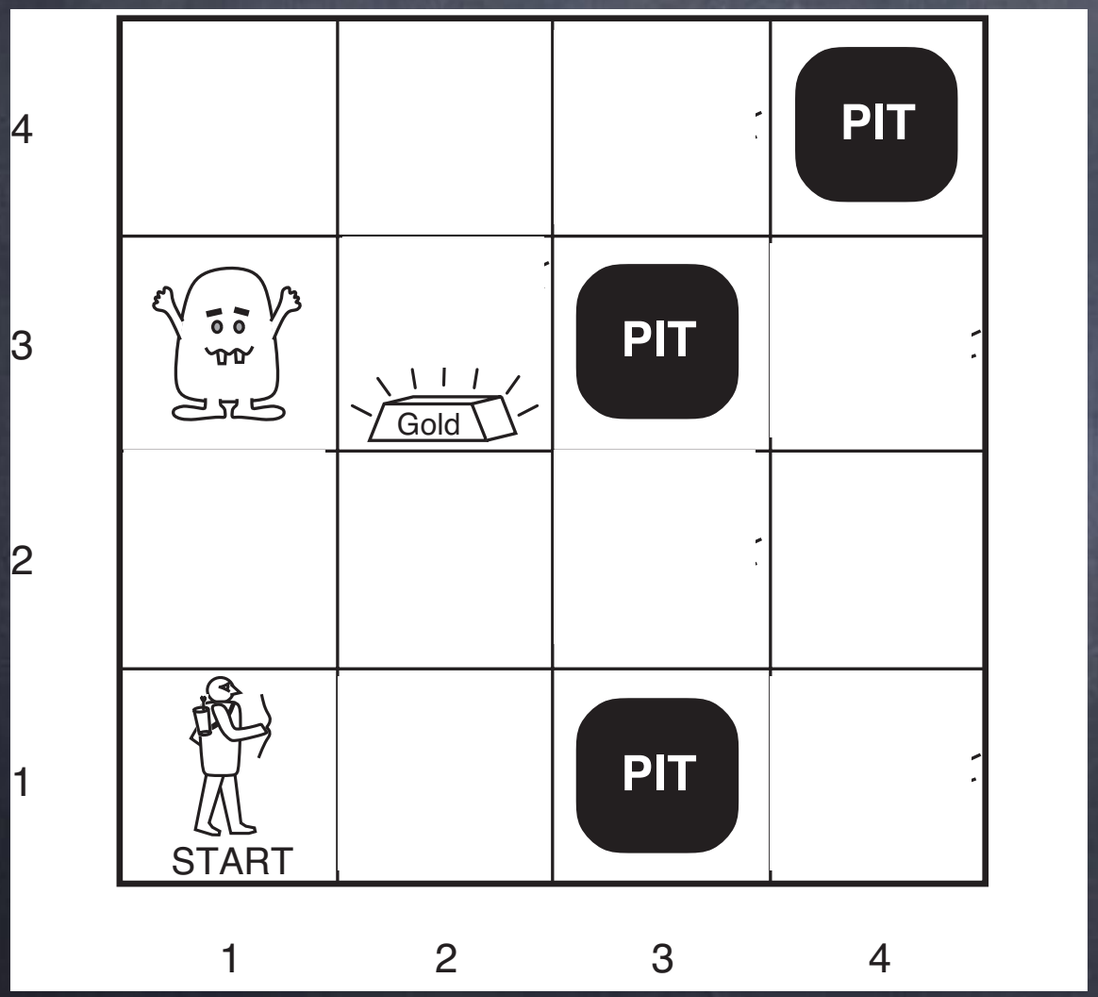
```
final int PIT = 0;
final int GOLD = 1;
final int WUMPUS = 2;
```

Representation



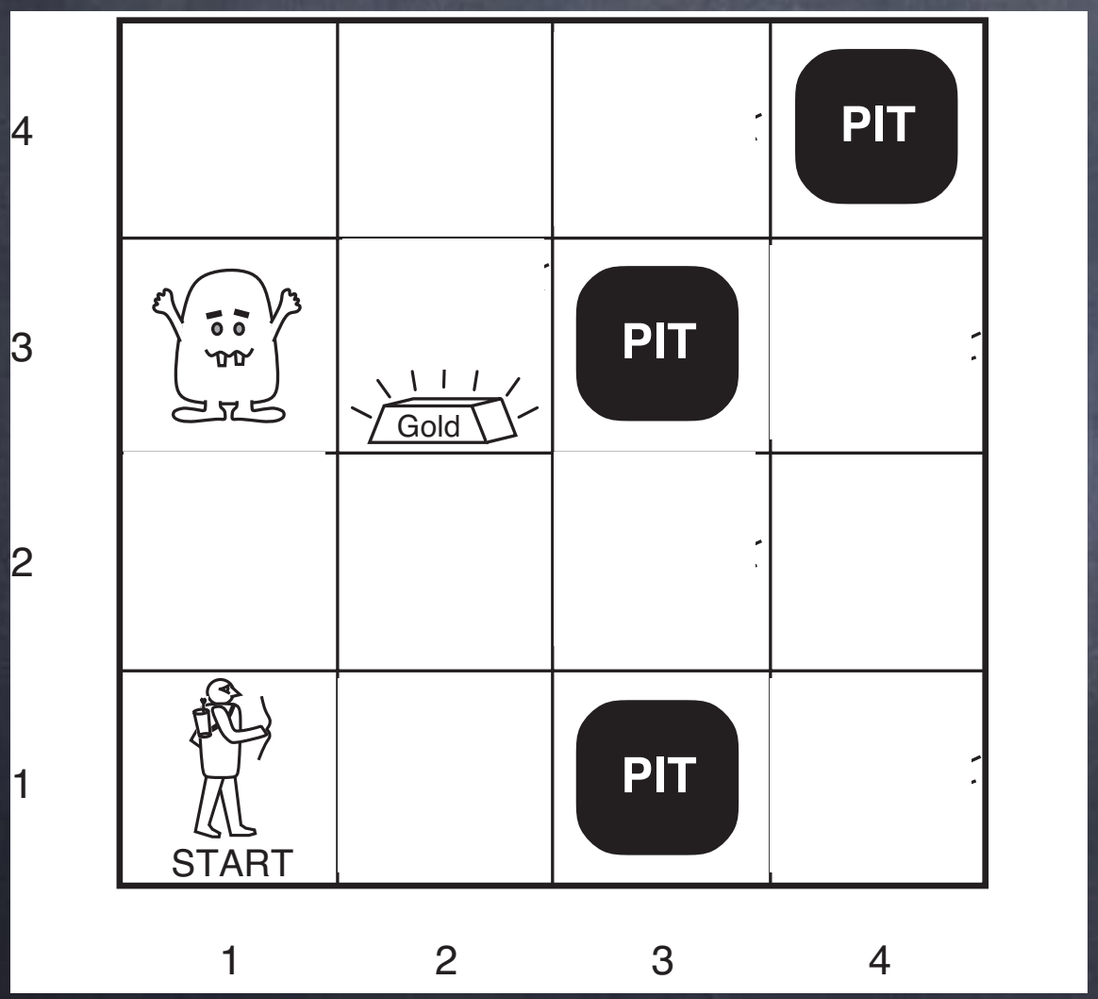

```
Boolean[][] P =
    new Boolean[n][n];
Boolean[][] G =
    new Boolean[n][n];
Boolean[][] W =
    new Boolean[n][n];

// P[i][j] == Boolean.TRUE
//    if there's a pit at [i,j]
// P[i][j] == Boolean.FALSE
//    if there's no pit at [i,j]
// P[i][j] == null
//    if we don't know
// G[i][j] ditto for gold
// W[i][j] ditto for wumpus
```

Representation



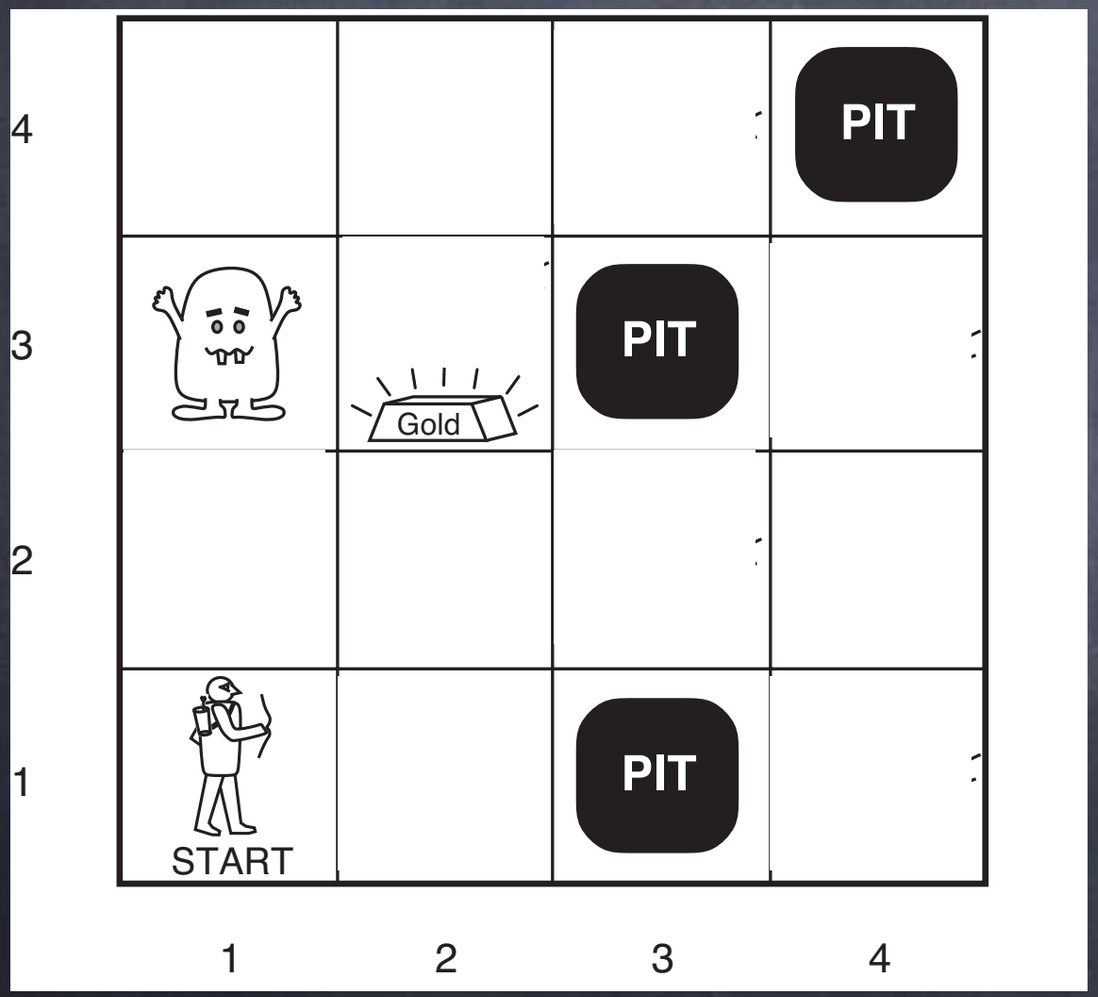

```
Position pos = <i, j>
```

Representation



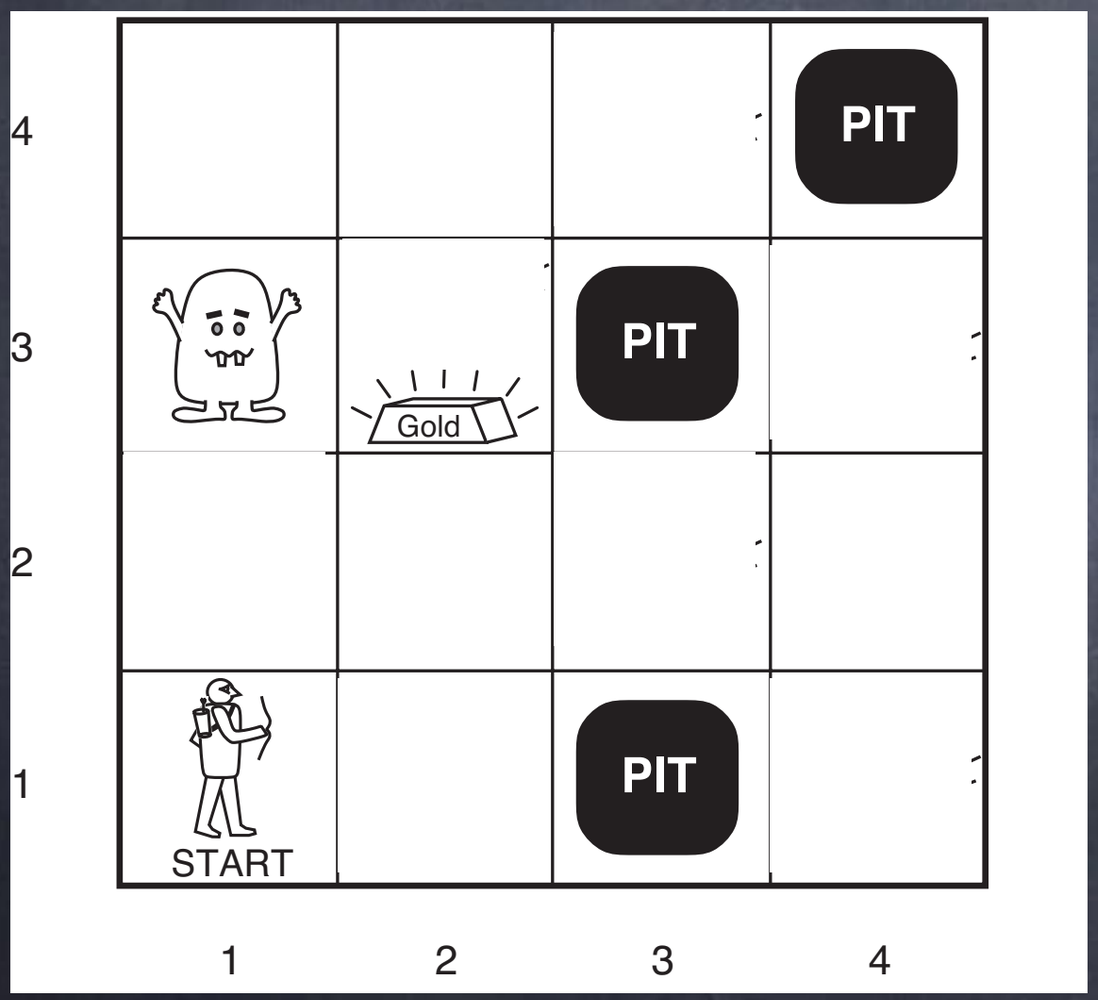

```
Boolean[][] At =
     new Boolean[n][n];

// At[i,j] == Boolean.TRUE
//    if agent is at location [i,j]
//    else Boolean.FALSE
```

Representation



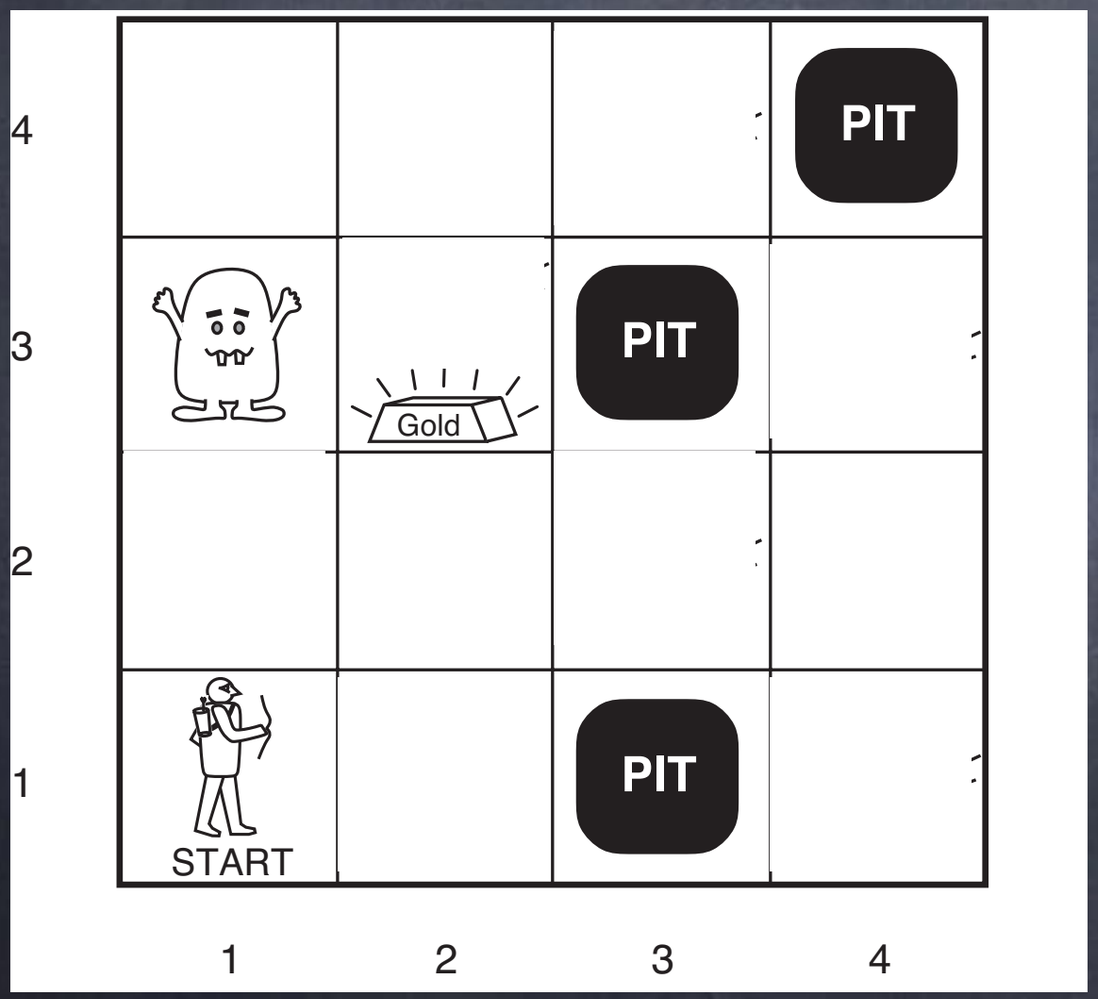

```
enum Orientation = { N,S,E,W };

Orientation orientation;
```

Representation



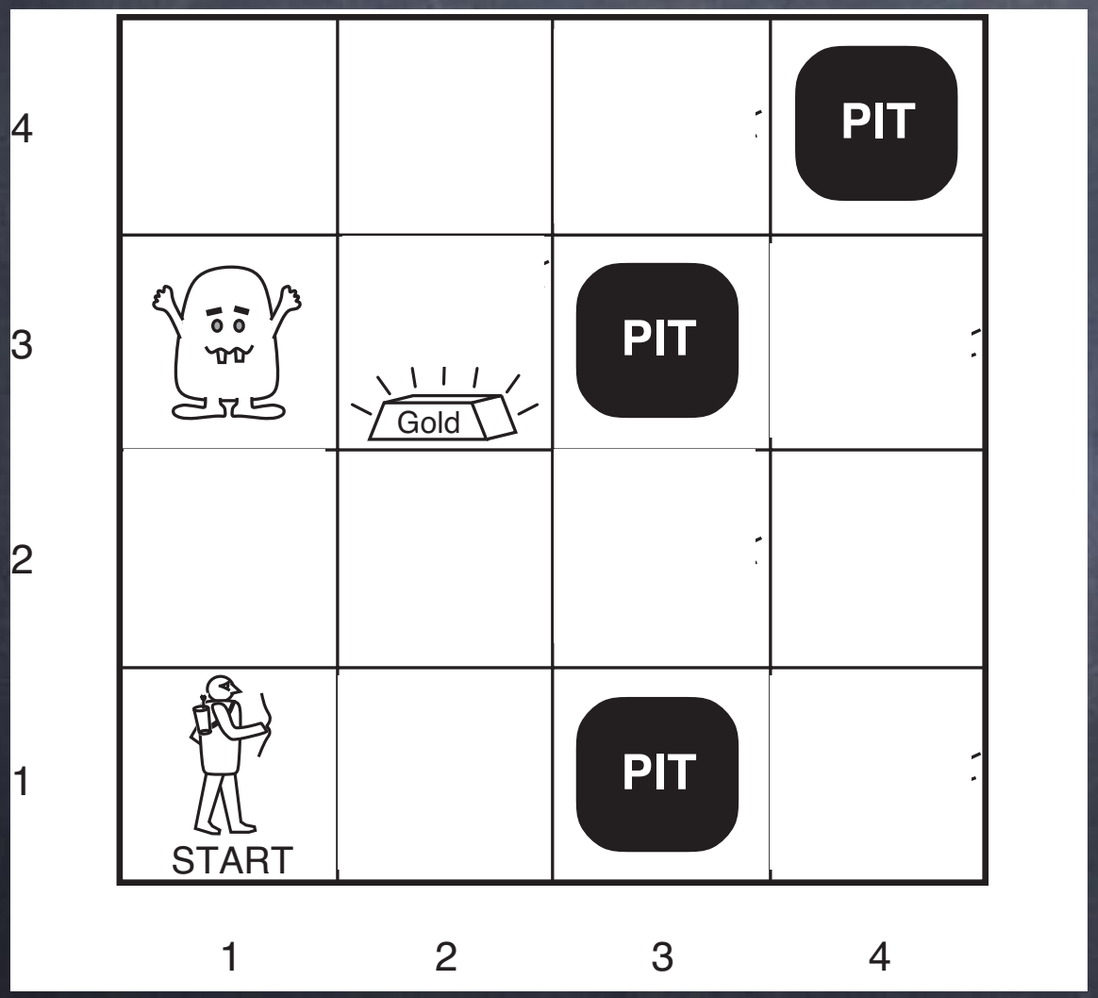

```
Boolean FacingN, FacingS,
    FacingE, FacingW;

FacingN == Boolean.TRUE
    if agent is facing north
...
```

Representation

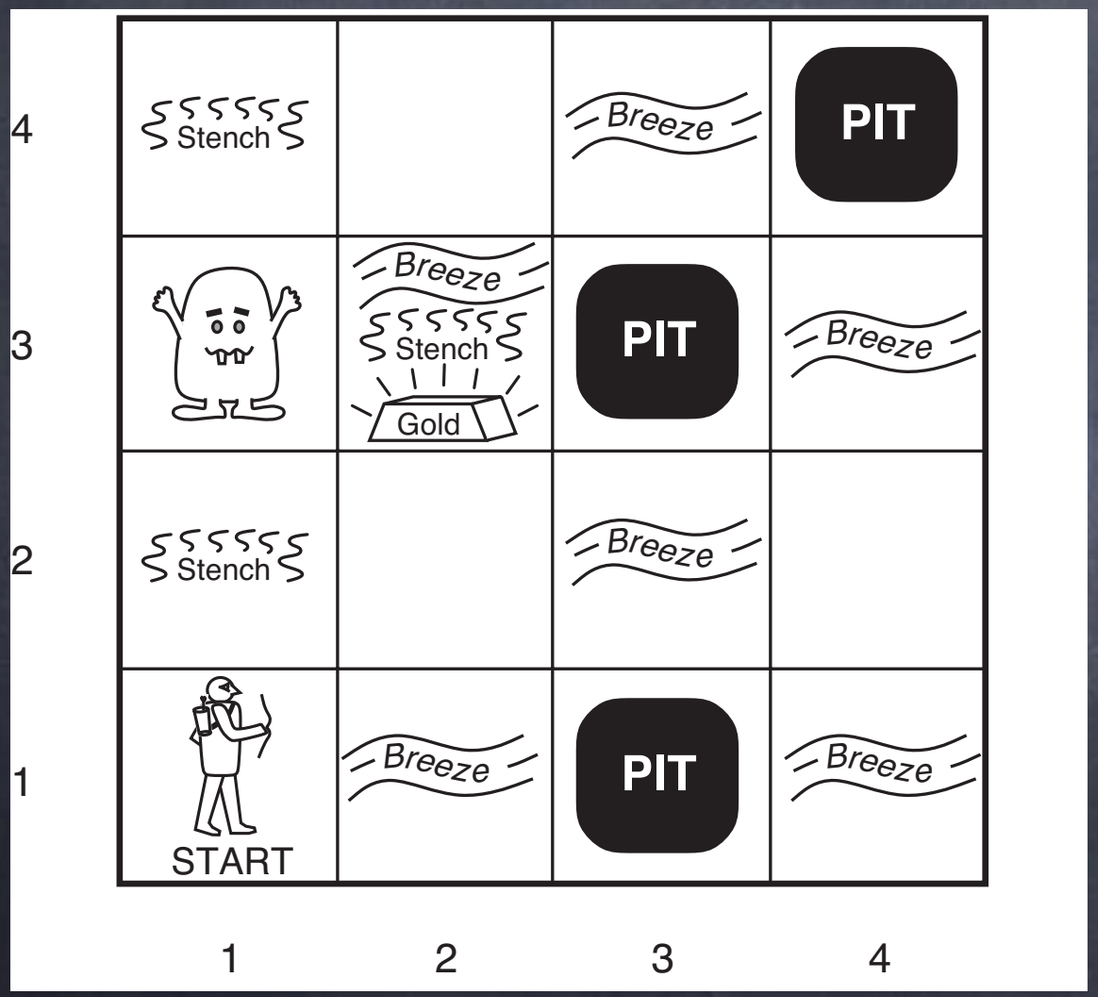



```
Boolean[][] P =
    new Boolean[n][n];
Boolean[][] G =
    new Boolean[n][n];
Boolean[][] W =
    new Boolean[n][n];
Boolean[][] At =
    new Boolean[n][n];
Boolean FacingN, FacingS,
    FacingE, FacingW;
```

Hunt The Wumpus

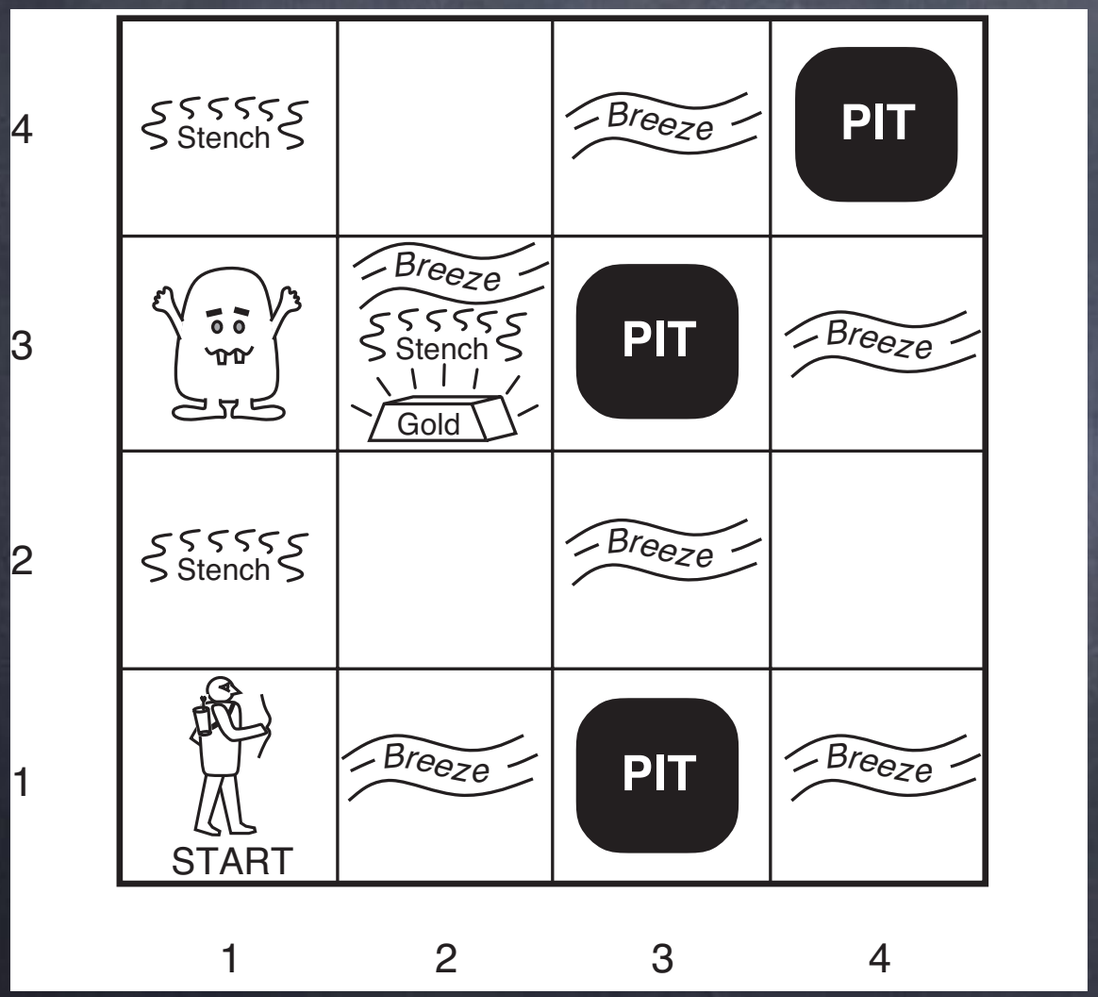


Representation



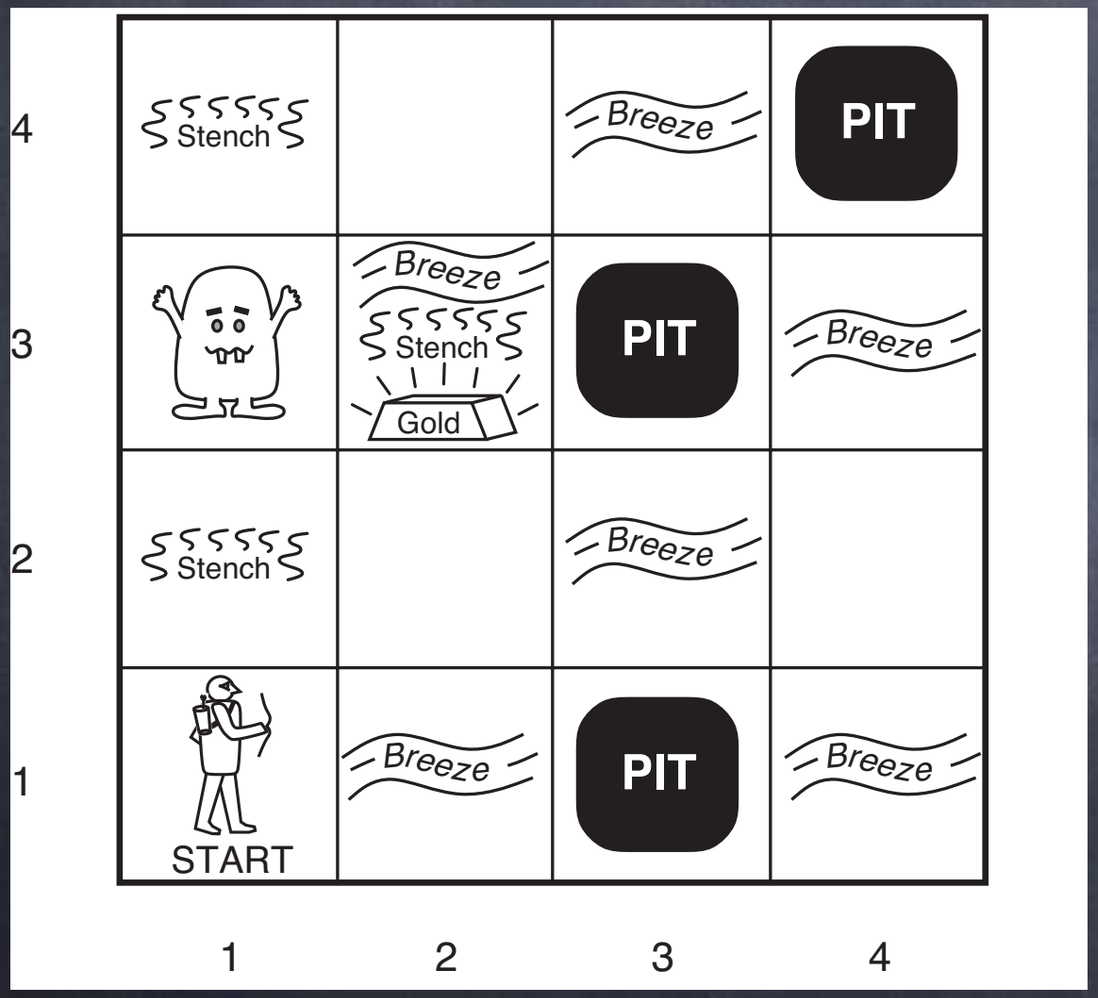

```
boolean[][][] sense =
    new boolean[n][n][5];

final int STENCH = 0;
final int BREEZE = 1;
final int GLITTER = 2;
final int BUMP = 3;
final int SCREAM = 4;
```

Representation



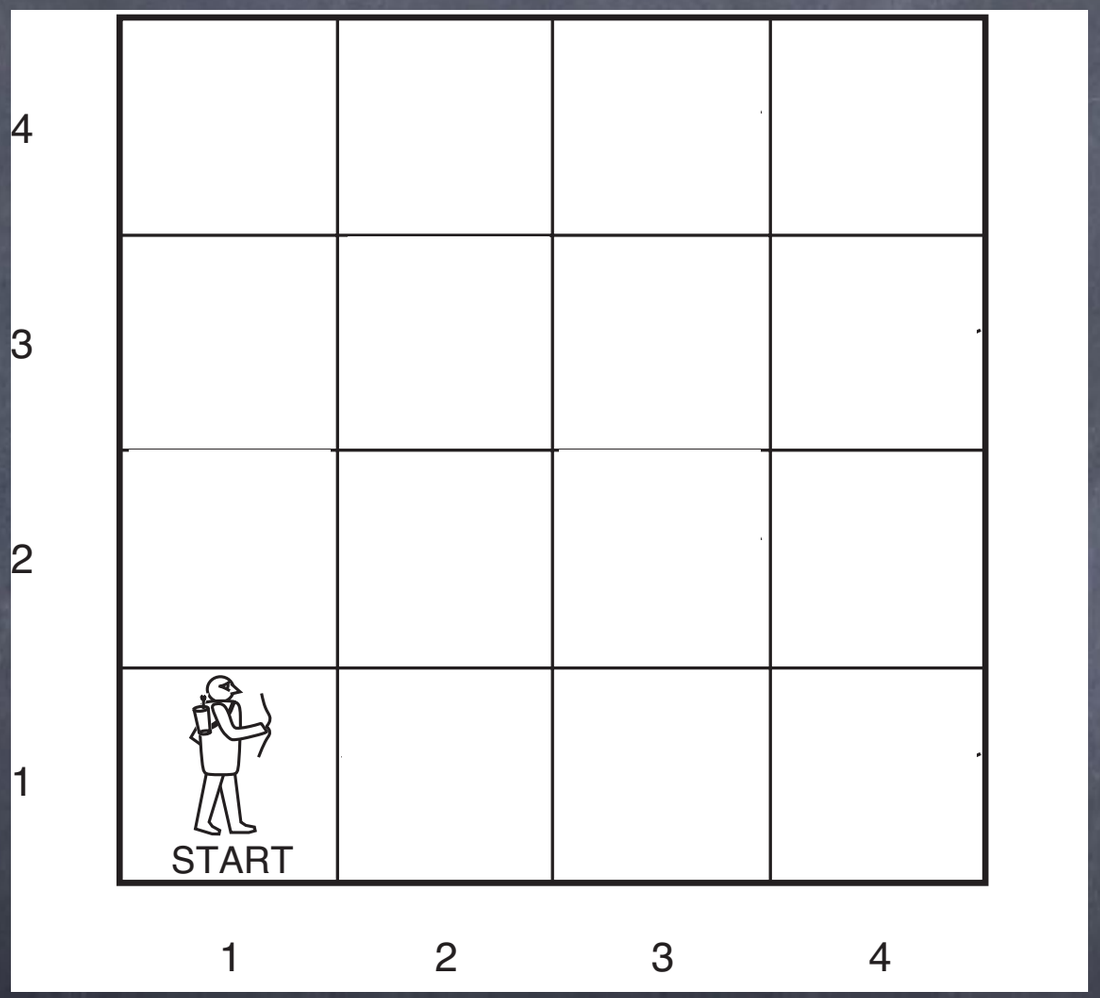

```
Boolean[][] S =
    new Boolean[n][n];
Boolean[][] B =
    new Boolean[n][n];
...

// S[i][j] == Boolean.TRUE
//    if the agent perceived a
//                  stench at [i,j]
// B[i][j] == Boolean.TRUE
//    if the agent perceived a
//                  breeze at [i,j]
...
```

Representation




```
Boolean[][] P =
    new Boolean[n][n];
Boolean[][] G =
    new Boolean[n][n];
Boolean[][] W =
    new Boolean[n][n];
Boolean[][] At =
    new Boolean[n][n];
Boolean FacingN, FacingS,
    FacingE, FacingW;
Boolean[][] S =
    new Boolean[n][n];
Boolean[][] B =
    new Boolean[n][n];
```

Hunt The Wumpus



1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
1,1 **A** **OK**	2,1	3,1	4,1

```
At[1][1] = true;  P[1][1] = false;
At[1][2] = false; W[1][1] = false;
...
At[2][1] = false;
At[2][2] = false;
...
```

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
1,1 **A** **OK**	2,1	3,1	4,1

```
At[1][1] = true;  P[1][1] = false;
At[1][2] = false; W[1][1] = false;
...               B[1][1] = false;
At[2][1] = false; S[1][1] = false;
At[2][2] = false;
...
```

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 **A** **OK**	2,1 **OK**	3,1	4,1

```
At[1][1] = true;  P[1][1] = false;
At[1][2] = false; W[1][1] = false;
...               B[1][1] = false;
At[2][1] = false; S[1][1] = false;
At[2][2] = false; W[1][2] = false;
...               P[1][2] = false;
                  W[2][1] = false;
                  P[2][1] = false;
```

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 **V** **OK**	2,1 **A** **OK**	3,1	4,1

```
At[1][1] = false; P[1][1] = false;     B[2][1] = true;
At[1][2] = false; W[1][1] = false;     S[2][1] = false;
...               B[1][1] = false;
At[2][1] = true;  S[1][1] = false;
At[2][2] = false; W[1][2] = false;
...               P[1][2] = false;
                  W[2][1] = false;
                  P[2][1] = false;
```

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** **OK**	2,1 **A** **B** **OK**	3,1 **P?**	4,1

```
At[1][1] = false; P[1][1] = false;     B[2][1] = true;
At[1][2] = false; W[1][1] = false;     S[2][1] = false;
...               B[1][1] = false;
At[2][1] = true;  S[1][1] = false;
At[2][2] = false; W[1][2] = false;
...               P[1][2] = false;
                  W[2][1] = false;
                  P[2][1] = false;
```

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **A** **OK**	2,2 P?	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 P?	4,1

```
At[1][1] = false;
At[1][2] = true;
...
At[2][1] = false;
At[2][2] = false;
...
```

```
P[1][1] = false;
W[1][1] = false;
B[1][1] = false;
S[1][1] = false;
W[1][2] = false;
P[1][2] = false;
W[2][1] = false;
P[2][1] = false;
```

```
B[2][1] = true;
S[2][1] = false;
B[1][2] = false;
S[1][2] = true;
W[1][3] = true;
P[2][2] = false;
P[3][1] = true;
```

1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 **S** **OK**	2,2 A **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

```
At[1][1] = false;
At[1][2] = false;
...
At[2][1] = false;
At[2][2] = true;
...
```

```
P[1][1] = false;
W[1][1] = false;
B[1][1] = false;
S[1][1] = false;
W[1][2] = false;
P[1][2] = false;
W[2][1] = false;
P[2][1] = false;
```

```
B[2][1] = true;
S[2][1] = false;
B[1][2] = false;
S[1][2] = true;
W[1][3] = true;
P[2][2] = false;
P[3][1] = true;
```

Background Knowledge

- General properties (or rules) of the world
- "In partially observable environments, combine general knowledge with current percepts to infer hidden aspects of the current state prior to selecting actions." (AIMA)

If you perceive a stench, then the wumpus is in an adjacent square.

If the wumpus is in an adjacent square, then you will perceive a stench.

If you perceive a stench, then the wumpus is in an adjacent square.

```
if S[i,j]
then (W[i-1,j] or W[i+1,j] or W[i,j-1] or W[i,j+1])
```

If the wumpus is in an adjacent square, then you will perceive a stench.

```
if (W[i-1,j] or W[i+1,j] or W[i,j-1] or W[i,j+1])
then S[i,j]
```

Warning: not a real programming language

Boolean CSP

- Variables describing attributes or features of the world
  - Factored representation
  - Domains: Boolean $\rightarrow$ { true, false}
  - Constraints: Identify possible combinations of the boolean variables

If you perceive a stench, then the wumpus is in an adjacent square.

```
if S[i,j]
then (W[i-1,j] or W[i+1,j] or W[i,j-1] or W[i,j+1])
```

If the wumpus is in an adjacent square, then you will perceive a stench.

```
if (W[i-1,j] or W[i+1,j] or W[i,j-1] or W[i,j+1])
then S[i,j]
```

Warning: not a real programming language

A Programming Language for Knowledge

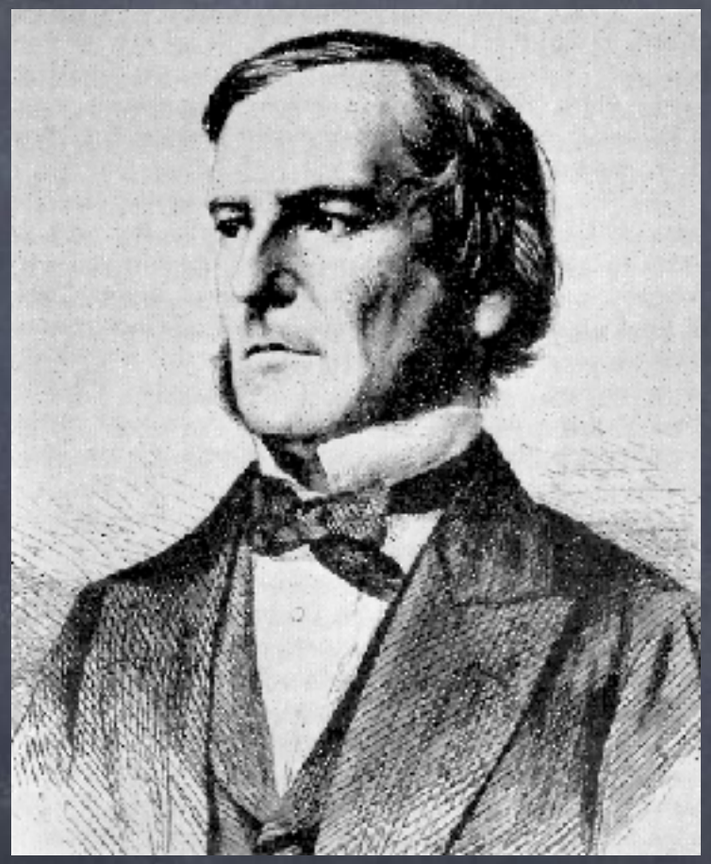
Knowledge: Constraints on how the world must be

A Programming Language for Knowledge

Aristole

(384BC - 332BC)

A Programming Language for Knowledge



George Boole

(1815-1864)

A Programming Language for Knowledge

Propositional Logic

Proposition

- Something that can be true or false
  - "it is raining"
  - "socrates is a woman"
  - "there is a wumpus at location 1,3 and I perceived a stench at location 1,2"
  - "either there is a pit at 2,2 or there is a wumpus at 2,2 and there is a pit at 2,3"

Atomic Proposition

- "it is raining"
- "socrates is a woman"
- "there is a wumpus at location 1,3"
- P, Q, R, ...

Boolean variables!

Boolean Algebra

- Operands
  - Propositions
  - (and variables that range over them)
- Operators

Connectives

- Connect propositions into larger propositions that can also be true or false
  - "there is a wumpus at location 1,3 and I perceived a stench at location 1,2"
  - "either there is a pit at 2,2 or there is a wumpus at 2,2 and there is a pit at 2,3"

Connectives

P is true	P
P is false ("not P")	$\neg P$
P and Q	$P \wedge Q$
P or Q	$P \vee Q$
if P then Q	$P \Rightarrow Q$
P if and only if Q	$P \Leftrightarrow Q$

Parentheses: "(" and ")"

Propositional Logic

- Syntax:
  - What counts as a well-formed statement, formula, sentence, or program
- Semantics:
  - What these statements, formulas, sentences, or programs mean

Semantics of Proposition Logic

P
F
T

Negation (not)

P	$\neg P$
F	T
T	F

Conjunction (and)

P	Q	$\neg P$	$P \wedge Q$
F	F		
F	T	T	
T	F		
T	T	F	

Conjunction (and)

P	Q	$\neg P$	$P \wedge Q$
F	F		F
F	T	T	F
T	F		F
T	T	F	T

Disjunction (or)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$
F	F		F	
F	T	T	F	
T	F		F	
T	T	F	T	

Disjunction (or)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$
F	F		F	F
F	T	T	F	T
T	F		F	T
T	T	F	T	

Disjunction (or)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$
F	F		F	F
F	T	T	F	T
T	F		F	T
T	T	F	T	T

Implication (if-then)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F		F	F	
F	T	T	F	T	
T	F		F	T	
T	T	F	T	T	

Implication (if-then)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F		F	F	
F	T	T	F	T	
T	F		F	T	
T	T	F	T	T	T

Implication (if-then)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F		F	F	
F	T	T	F	T	
T	F		F	T	F
T	T	F	T	T	T

Implication (if-then)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F		F	F	T
F	T	T	F	T	T
T	F		F	T	F
T	T	F	T	T	T

Biconditional (if and only if)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F		F	F	T	
F	T	T	F	T	T	
T	F		F	T	F	
T	T	F	T	T	T	

Biconditional (if and only if)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F		F	F	T	
F	T	T	F	T	T	
T	F		F	T	F	
T	T	F	T	T	T	T

Biconditional (if and only if)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F		F	F	T	T
F	T	T	F	T	T	
T	F		F	T	F	
T	T	F	T	T	T	T

Semantics of Propositional Logic

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F		F	F	T	T
F	T	T	F	T	T	F
T	F		F	T	F	F
T	T	F	T	T	T	T

You perceive a breeze at [1,1] if and only if there is a pit at [2,1] or there is a pit at [1,2]

You perceive a breeze at [1,1] if and only if there is a pit at [2,1] or there is a pit at [1,2]

P: You perceive a breeze at [1,1]

Q: There is a pit at [1,2]

R: There is a pit at [2,1]

You perceive a breeze at [1,1] if and only if there is a pit at [2,1] or there is a pit at [1,2]

$B_{1,1}$: You perceive a breeze at [1,1]

$P_{1,2}$: There is a pit at [1,2]

$P_{2,1}$: There is a pit at [2,1]

You perceive a breeze at [1,1] if and only if there is a pit at [2,1] or there is a pit at [1,2]

$B_{1,1}$: You perceive a breeze at [1,1]

$P_{1,2}$: There is a pit at [1,2]

$P_{2,1}$: There is a pit at [2,1]

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

Truth Table

Truth Table

$B_{1,1}$	$P_{1,2}$	$P_{2,1}$

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

Truth Table

$B_{1,1}$	$P_{1,2}$	$P_{2,1}$
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	T	F
T	F	T
T	T	T

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

Truth Table

$B_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{1,2} \vee P_{2,1}$
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	F
T	T	F	T
T	F	T	T
T	T	T	T

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

Truth Table

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

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

Propositional Logic (So Far)

- Syntax: how to write out legal statements in the language
- Semantics: whether a statement is true or false given the truth/falsity of the atomic propositions
- Application: Representing knowledge (e.g., of the Wumpus world)

THE OFFICIAL

METALLICA

ILLUSTRATED CHRONICLES

SO WHAT!

THE GOOD, THE MAD AND THE UGLY

EDITED BY STEFFAN CHIRAZI

If you perceive a stench, then the wumpus is in an adjacent square.

$$S_{1,1} \Rightarrow (W_{1,2} \vee W_{2,1})$$

$$S_{1,2} \Rightarrow (W_{1,1} \vee W_{2,2} \vee W_{1,3})$$

$$S_{2,1} \Rightarrow (W_{1,1} \vee W_{2,2} \vee W_{3,1})$$

$$S_{2,2} \Rightarrow (W_{1,2} \vee W_{2,1} \vee W_{2,3} \vee W_{3,2})$$

If the wumpus is in an adjacent square, then you will perceive a stench.

$$(W_{1,2} \vee W_{2,1}) \Rightarrow S_{1,1}$$

$$(W_{1,1} \vee W_{2,2} \vee W_{1,3}) \Rightarrow S_{1,2}$$

$$(W_{1,1} \vee W_{2,2} \vee W_{3,1}) \Rightarrow S_{2,1}$$

$$(W_{1,2} \vee W_{2,1} \vee W_{2,3} \vee W_{3,2}) \Rightarrow S_{2,2}$$

$$At_{1,1}$$

$$\neg S_{1,1}$$

$$At_{1,1}$$

$$\neg S_{1,1}$$

$$S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1})$$

$$At_{1,1}$$

$$\neg S_{1,1}$$

$$S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1})$$

$$\neg W_{1,2} \ ?$$

$$\neg W_{2,1} \ ?$$

Inference

- Given a set of statements (facts and background knowledge)
- Determine whether some other statement is true
- Makes implicit information explicit

Model (Possible World)

- Assignment of true or false to all the propositional variables

Model (Possible World)

- Assignment of true or false to all the propositional variables
- A model satisfies a sentence if it makes the sentence true
- "A model of the sentence"

Satisfaction

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

Satisfaction

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

Inference

- Given a set of statements (facts and background knowledge)
- Determine whether some other statement is true
- Makes implicit information explicit

Entailment

- If α entails β: $\alpha \models \beta$
  - Whenever α is true, so is β

Entailment

- If α entails β: $\alpha \models \beta$
  - Whenever α is true, so is β
  - Every model of α is a model of β
  - Models(α) $\subseteq$ Models(β)
  - α is at least as strong an assertion as β
    - Rules out no fewer possible worlds

Entailment

- If α entails β: $\alpha \models \beta$
  - Whenever α is true, so is β
  - Every model of α is a model of β
  - Models(α) $\subseteq$ Models(β)
  - α is at least as strong an assertion as β
    - Rules out no fewer possible worlds

Entailment

- If α entails β: $\alpha \models \beta$
  - Whenever α is true, so is β
  - Every model of α is a model of β
    - Models(α) $\subseteq$ Models(β)
    - α is at least as strong an assertion as β
      - Rules out no fewer possible worlds

Entailment

P	Q	...	α	β
F	F	...	F	T
F	T	...	F	F
F	F	...	F	F
F	T	...	T	T
T	F	...	T	T
T	T	...	T	T
T	F	...	F	F
T	T	...	F	T

Entailment

P	Q	...	α	β
F	F	...	F	T
F	T	...	F	F
F	F	...	F	F
F	T	...	T	T
T	F	...	T	T
T	T	...	T	T
T	F	...	F	F
T	T	...	F	T

Model Checking

P	Q	...	α	β
F	F	...	F	T
F	T	...	F	F
F	F	...	F	F
F	T	...	T	T
T	F	...	T	T
T	T	...	T	T
T	F	...	F	F
T	T	...	F	T

Inference

- If α entails β
  - Then when you know α is true you also know β is true
    - Even though you didn't have it written down down explicitly!
- α is your knowledge, β your question

Entailment

- If α entails β: $\alpha \models \beta$
  - Whenever α is true, so is β
  - Every model of α is a model of β
    - Models(α) $\subseteq$ Models(β)
    - α is at least as strong an assertion as β
      - Rules out no fewer possible worlds

$O(2^n)$

P	Q	...	α	β
F	F	...	F	T
F	T	...	F	F
F	F	...	F	F
F	T	...	T	T
T	F	...	T	T
T	T	...	T	T
T	F	...	F	F
T	T	...	F	T

Computing Entailment

```
boolean tt_entails(Set<Sentence> kb, Sentence alpha) {
    List<Symbol> symbols = new List(kb.getSymbols());
    symbols.append(alpha.getSymbols());
    return tt_check_all(kb, alpha, symbols, new Model());
}

boolean tt_check_all(Set<Sentence> kb, Sentence alpha,
                     List<Symbol> symbols, Model model) {
    if (symbols.isEmpty()) {
        if (model.satisfies(kb)) {
            return model.satisfies(alpha);
        } else {
            return true;
        }
    } else {
        Symbol p = symbols.removeFirst();
        return (tt_check_all(kb, alpha, symbols,
                             model.clone().assign(p, Boolean.TRUE)) &&
                tt_check_all(kb, alpha, symbols,
                             model.clone().assign(p, Boolean.FALSE)));
    }
}
```

Propositional Logic

- Programming language for knowledge
- Factored representation of state
  - Propositions, connectives, sentences
- Model (possible world) = assignment of true or false to all the propositions
- Entailment: $\alpha \models \beta$
  - Every model of α is a model of β

For next time:

AIMA Ch. 7.5