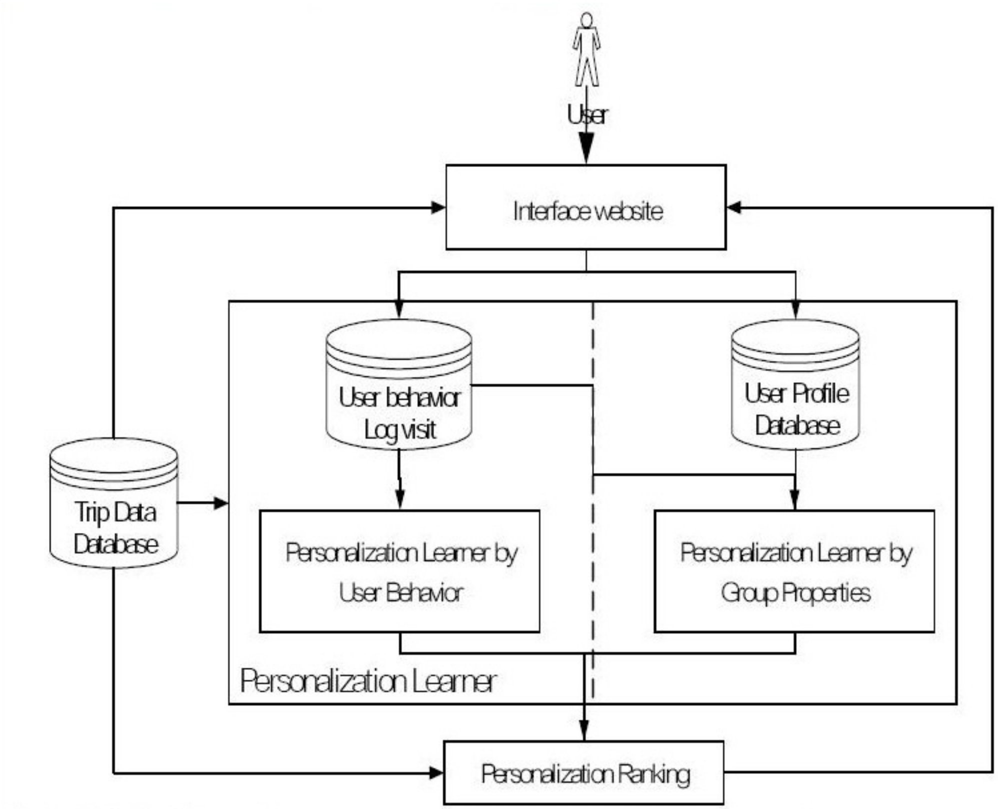
Intelligent Agents

Intelligent agents

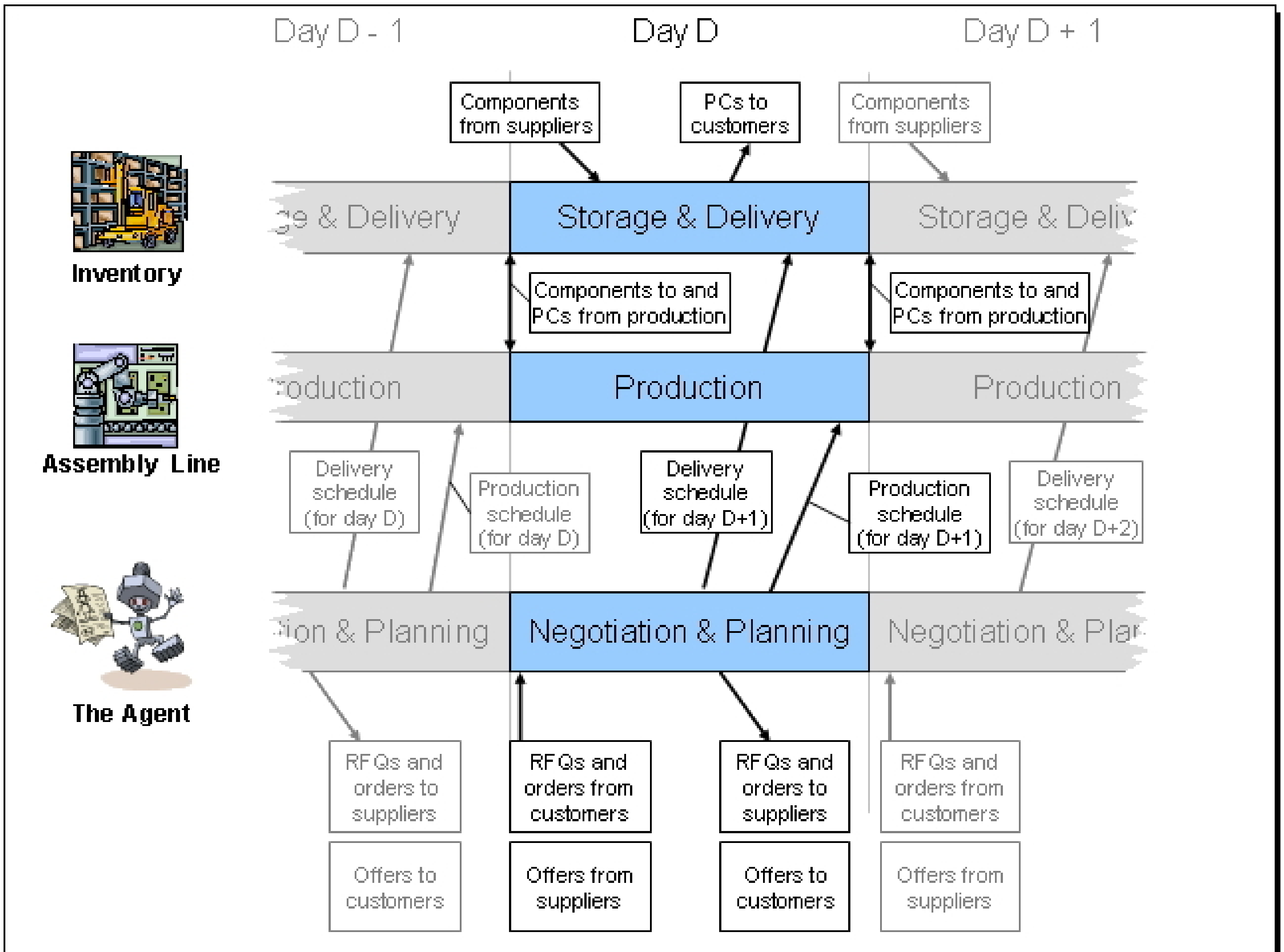
- are autonomous entities
- observe and act in an environment
- exhibit goal-oriented behavior

Example: Personal Software Agents



Virtual Reality Agents

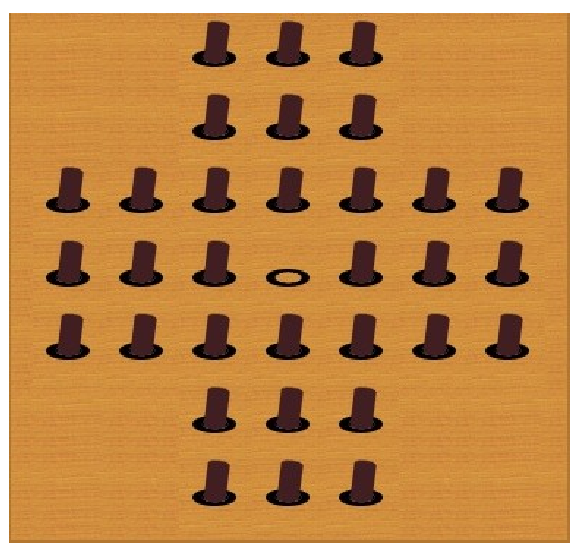
Trading Agents



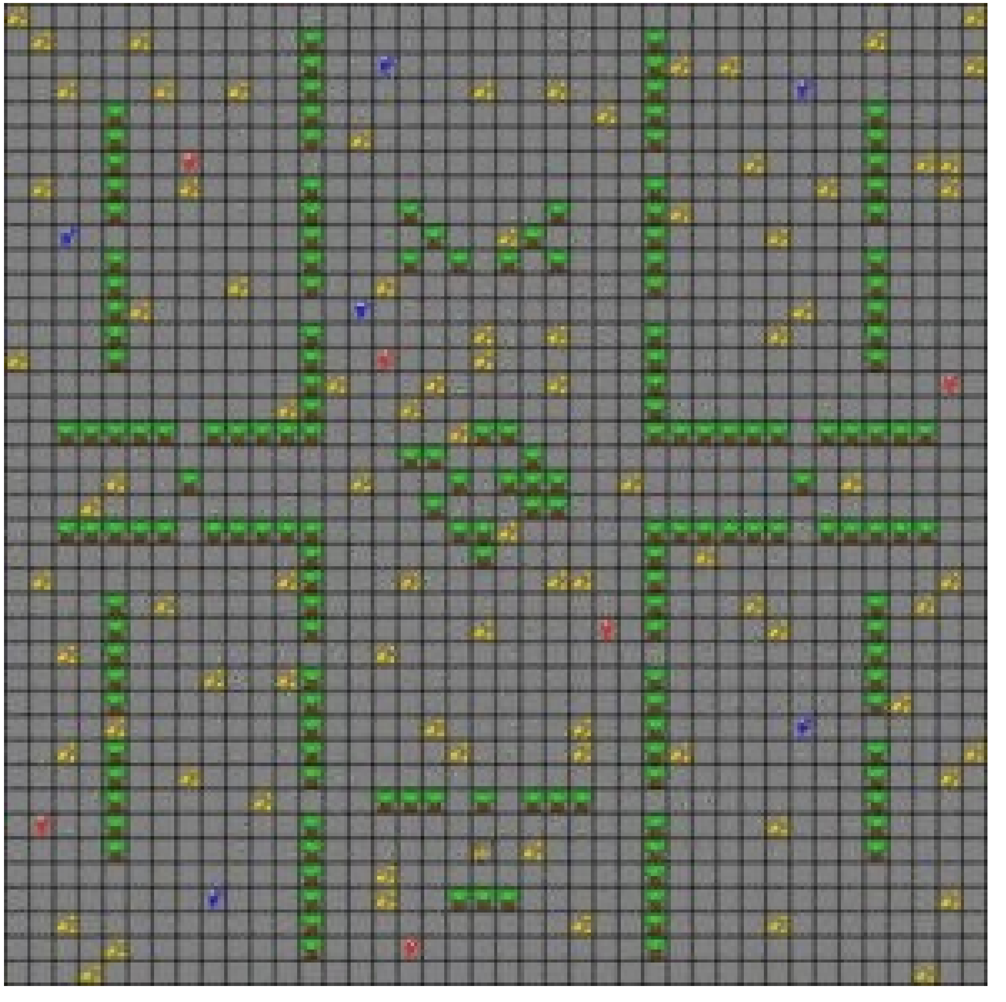
General Game Playing Agents

A general game playing agent

- understands the rules of arbitrary games
- learns to play these games well without human intervention

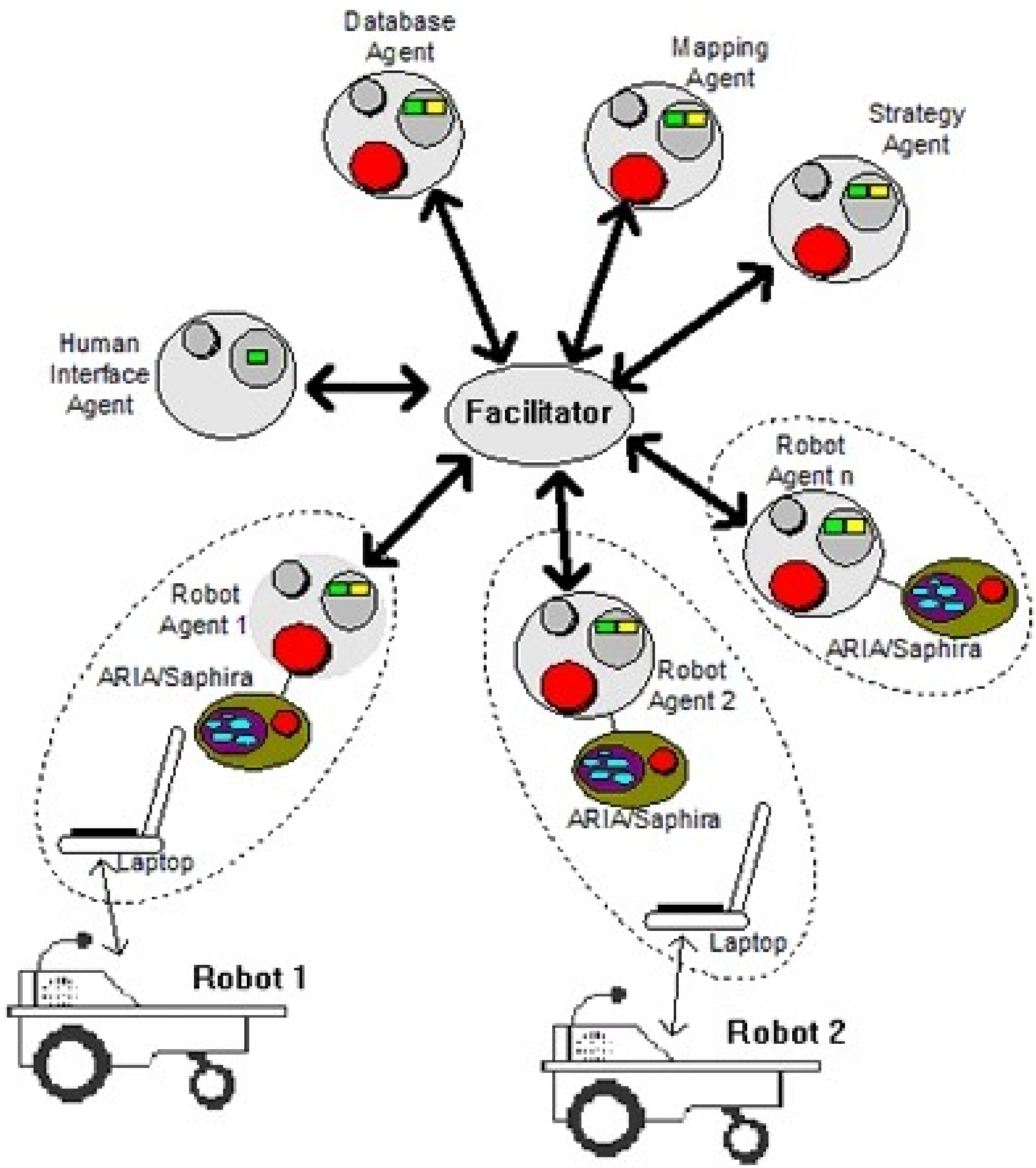


Multi-Agent Systems (I)



PROMAS Agent Contest 2007

Multi-Agent Systems (II)



Reactive Agents

RoboCup Soccer: Simulation League

Robotic Agents (I)

RoboCup Soccer

Robotic Agents (II)

Museum Guide RHINO

Schedule

April, 15	Introduction
April, 20, 27 & 29	Procedural Action Programs (GOLOG)
May, 4 & 11	Planning
May, 18 & 25	Declarative Action Programs
May, 27 & June, 8	Commonsense Reasoning for Agents
June, 15, 17 & 22	Reactive Action Programs (Jason)
June, 29	Multi-Agent Systems
July, 1	Robotics
July, 6	Wrap Up

Preliminaries: Logic

- Atoms are predicate symbols with terms as arguments

- Formulas are built up from atoms and the connectives
  - $\forall$ (universal quantification)
  - $\exists$ (existential quantification)
  - $\neg$ (negation)
  - $\wedge$ (conjunction)
  - $\vee$ (disjunction)
  - $\supset$ (implication)
  - $\equiv$ (equivalence)

- Variables outside a quantifier are always universally quantified

Preliminaries: Equality

The binary predicate symbol "=" shall always be interpreted as the identity relation.

Note: This is equivalent to adding the standard axioms of equality in classical first-order logic

- $x = x$
- $x = y \supset y = x$
- $x = y \wedge y = z \supset x = z$
- $x_i = y \supset f(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, y, \ldots, x_n)$
- $x_i = y \supset [P(x_1, \ldots, x_i, \ldots, x_n) \equiv P(x_1, \ldots, y, \ldots, x_n)]$

Preliminaries: Sorts (I)

Sorts are used to define the range of the arguments of predicates and functions.

Standard sorts are $\mathbb{N}$ and $\mathbb{R}$.

- a variable $x : S$ is a term of sort S
- if $f : S_1 \times ... \times S_n \mapsto S$ and $t_1 , ... , t_n$ terms of sort $S_1 , ... , S_n$ then $f(t_1 , ... , t_n)$ is a term of sort S
- if $P : S_1 \times ... \times S_n$ and $t_1 , ... , t_n$ terms of sort $S_1 , ... , S_n$ then $P(t_1 , ... , t_n)$ is a well-sorted atom

Standard functions and predicates are $+ : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$ and $\leq : \mathbb{R} \times \mathbb{R}$ etc.

Preliminaries: Sorts (II)

An interpretation for a logic with sorts consists of

- a non-empty domain D_i for each sort S_i
- a mapping $D_i \times ... \times D_j \mapsto D_k$ for each $f : S_i \times ... \times S_j \mapsto S_k$
- a relation $D_i \times ... \times D_j$ for each $P : S_i \times ... \times S_j$

Note: This is equivalent to classical first-order logic with

- a unary predicate for each sort

- conditioning all quantified sub-formulas to be well-sorted

Preliminaries: *UNA*

The unique-name-assumption says that different symbols mean different things.

Formally, *UNA* [h_1, ..., h_n] abbreviates the formula

$$\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} h_i(\vec{x}) \neq h_j(\vec{y}) \wedge \bigwedge_{i=1}^{n} \left[h_i(\vec{x}) = h_i(\vec{y}) \supset \vec{x} = \vec{y} \right]$$

Example: *UNA* [*Location*, *Blue*] implies *Location* (3) $\neq$ *Blue* and *Location* (3) $\neq$ *Location* (4) (given that 3 $\neq$ 4)

Preliminaries: Logic Programs

```
terminal:- line(xsymb).
terminal:- line(osymb).
terminal:- not open.

line(S):- row(S).
line(S):- col(S).
line(S):- diag(S).

row(X):- holds(cell(M,1,X)), holds(cell(M,2,X)), holds(cell(M,3,X)).
col(X):- holds(cell(1,N,X)), holds(cell(2,N,X)), holds(cell(3,N,X)).
diag(X):- holds(cell(1,1,X)), holds(cell(2,2,X)), holds(cell(3,3,X)).
diag(X):- holds(cell(3,1,X)), holds(cell(2,2,X)), holds(cell(1,3,X)).

open:- holds(cell(M,N,blank)).
```

Preliminaries: Completion Semantics

Take all clauses for a predicate *P*

$$P(\vec{t}_1) :- L_{11}, \ldots, L_{1n_1}$$

$$\vdots$$

$$P(\vec{t}_m) :- L_{m1}, \ldots, L_{mn_m}$$

The logical definition for *P* is

$$P(\vec{x}) \equiv \bigvee_{i=1}^{m} (\exists \vec{y}_i)(\vec{x} = \vec{t}_i \land L_{i1} \land \ldots \land L_{in_i})$$

The completion *COMP*[*P*] of logic program *P* consists of

- the logical definitions for each predicate
- the unique-name-assumption for all function symbols (incl. constants)

Completion Semantics: Example

$$
\begin{aligned}
\mathit{Terminal} &\equiv \mathit{Line}(\mathit{XSymb}) \vee \mathit{Line}(\mathit{OSymb}) \vee \neg \mathit{Open} \\
\mathit{Line}(s) &\equiv \mathit{Row}(s) \vee \mathit{Col}(s) \vee \mathit{Diag}(s) \\
\mathit{Row}(x) &\equiv (\exists m)\,(\mathit{Holds}(\mathit{Cell}(m, 1, x)) \wedge \mathit{Holds}(\mathit{Cell}(m, 2, x)) \wedge \mathit{Holds}(\mathit{Cell}(m, 3, x))) \\
\mathit{Col}(x) &\equiv (\exists n)\,(\mathit{Holds}(\mathit{Cell}(1, n, x)) \wedge \mathit{Holds}(\mathit{Cell}(2, n, x)) \wedge \mathit{Holds}(\mathit{Cell}(3, n, x))) \\
\mathit{Diag}(x) &\equiv \mathit{Holds}(\mathit{Cell}(1, 1, x)) \wedge \mathit{Holds}(\mathit{Cell}(2, 2, x)) \wedge \mathit{Holds}(\mathit{Cell}(3, 3, x)) \\
&\quad \vee \\
&\quad \mathit{Holds}(\mathit{Cell}(3, 1, x)) \wedge \mathit{Holds}(\mathit{Cell}(2, 2, x)) \wedge \mathit{Holds}(\mathit{Cell}(1, 3, x)) \\
\mathit{Open} &\equiv (\exists m, n)\,\mathit{Holds}(\mathit{Cell}(m, n, \mathit{Blank}))
\end{aligned}
$$

Linear Resolution Step

Given:

Query $L_1 \wedge L_2 \wedge ... \wedge L_m$ (without negation)

Clauses (without negation)

Let:

$A <= B_1 \wedge ... \wedge B_n$ "fresh" variant of a clause

σ mgu of L_1 and A

Then $L_1 \wedge L_2 \wedge ... \wedge L_m \rightarrow (B_1 \wedge ... \wedge B_n \wedge L_2 \wedge ... \wedge L_m)\sigma$

is a linear resolution step.

Derivation

A sequence of resolution steps is called a derivation.

A successful derivation ends with the empty query.

The answer substitution (inferred by a successful derivation)
is obtained by composing the mgu's $\sigma_1 \circ ... \circ \sigma_n$ of each step
(and restricting the result to the variables in the original query).

A failed derivation ends with a query to which no clause applies.

Linear Resolution with Negation

Given:

Query $L_1 \wedge L_2 \wedge ... \wedge L_m$

Clauses

- If L_1 is an atom, proceed as before
- If L_1 is of the form $\neg A$ and A is variable-free then
  - if all derivations for A fail then
    $L_1 \wedge L_2 \wedge ... \wedge L_m \rightarrow L_2 \wedge ... \wedge L_m$
  - if there is a successful derivation for A then
    $L_1 \wedge L_2 \wedge ... \wedge L_m \rightarrow$ fail

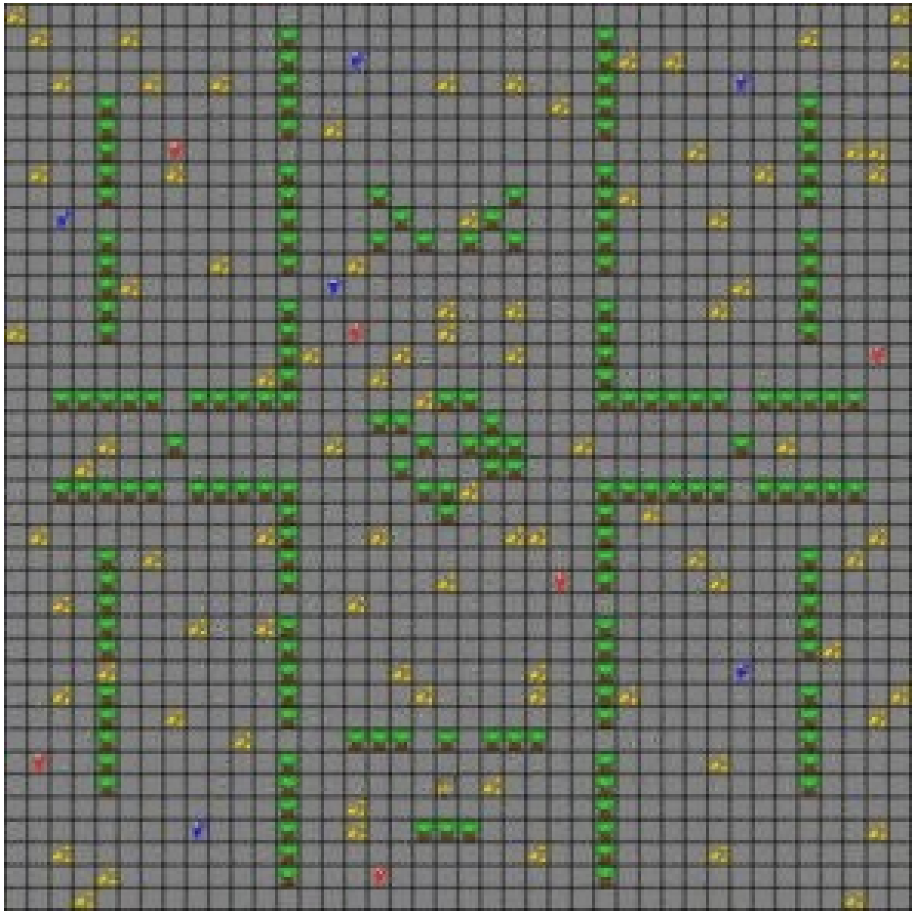
Derivations: Example

Consider the following facts in addition to the program on Slide 17:

```
holds(cell(1,1,xsymb)). holds(cell(1,2,osymb)). holds(cell(1,2,xsymb)).
holds(cell(2,1,blank)). holds(cell(2,2,osymb)). holds(cell(2,3,xsymb)).
holds(cell(3,1,osymb)). holds(cell(3,2,blank)). holds(cell(3,3,xsymb)).
```

Then there is a successful derivation for the query `terminal.`

Symbolic Representations of the Environment/Actions



Actions

Ship

Move(direction)

Pick

Drop

Mark

Unmark

Percepts

absolute-position

surrounding-cells

Symbolic Representations (II)

Fluents

Cell(*x*, *y*, *contents*)

Control(*player*)

CanCastle(*side*)

EnPassant(*file*)

Actions

Move(x_1, y_1, x_2, y_2)

Promote(y_1, y_2, *piece*)

Noop

Symbolic Representations (III)

High-level action

Go(*L*)

Low-level actions

PanTiltSetTrackPoint(*x*, *y*)

SpeechTalkText(*t*)

RobotDrivePath(*p*)

RobotTurnToPoint(*x*, *y*)

The Symbol Grounding Problem

What symbols are suitable to model a given domain?

How do symbols – e.g. “ball”, “goal”, “goalie” – relate to the real word?

Symbol anchoring is the problem of creating and maintaining the correspondence between symbols (i.e. abstract representations) and perceptual data that refer to the same object.