Multi-Agent Systems

Sept. 2000, Bahia Blanca

University Nacional del Sur

- **Last two weeks in September**.
- **Tentative Dates:** Tuesday, Sept. 19th, Thursday, Sept. 21st, Friday, Sept. 22nd, Tuesday, Sept. 26th, Thursday, Sept. 28th, Friday, Sept. 29th.
- **Time:** From 4–6 pm, unless otherwise indicated.
- Lecture Course is on theoretical issues, emphasis on mathematical-logical foundations.

Overview

1. Introduction, Terminology
2. Three Basic Architectures
3. Logic Based Architectures
4. Distributed Decision Making
5. Contract Nets, Coalition Formation

Chapter 3. Logic Based Architectures

3.1 Sentential Logic

3.2 Situation Calculus

3.3 Problems

3.4 A Solution to the Frame Problem?

3 Logic Based Architectures

Symbolic AI: Symbolic representation, e.g. sentential or first order logic. Using deduction. **Agent as a theorem prover**.

Traditional: Theory about agents. Implementation as stepwise process (Software Engineering) over many abstractions.

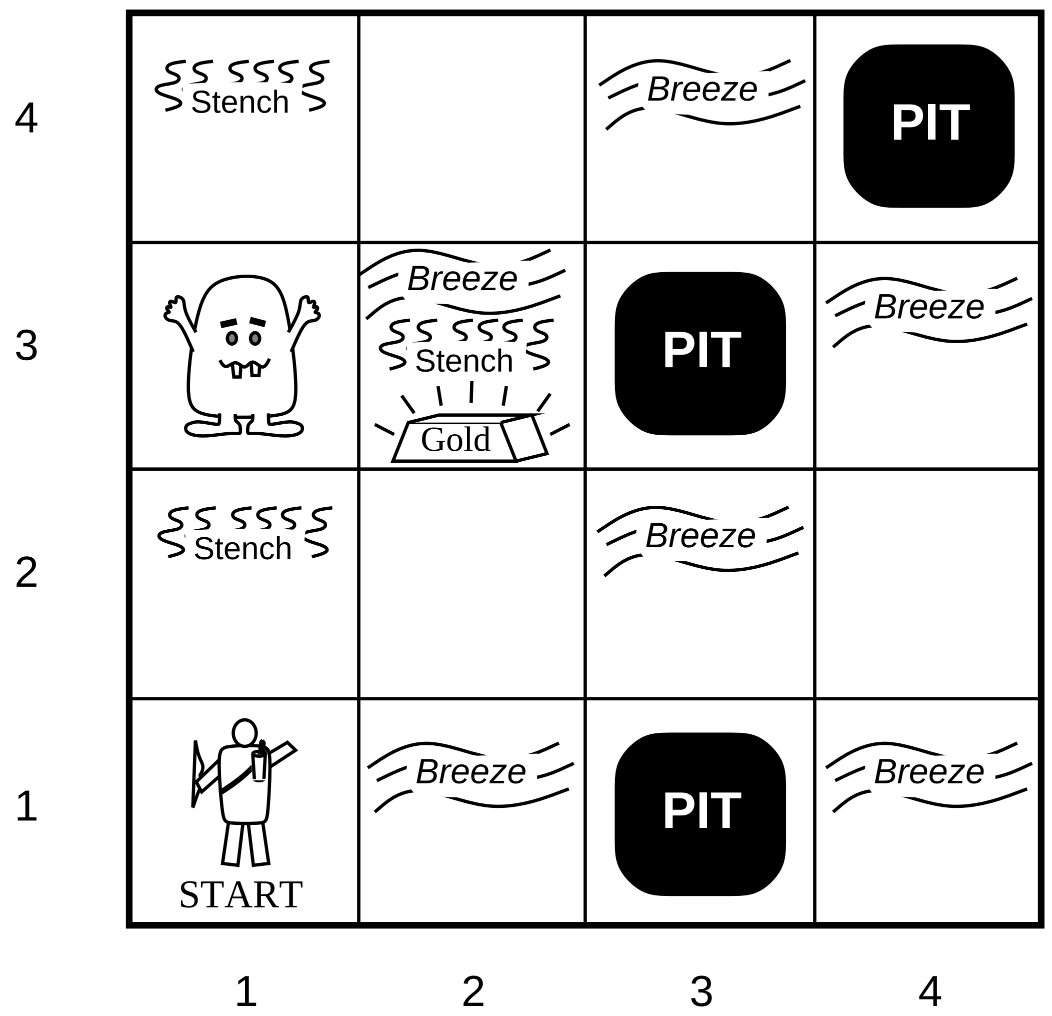
Symbolic AI: View the theory itself as **executable specification**.
Internal state: *Knowledge Base* (KB), often simply called **D** (**database**).

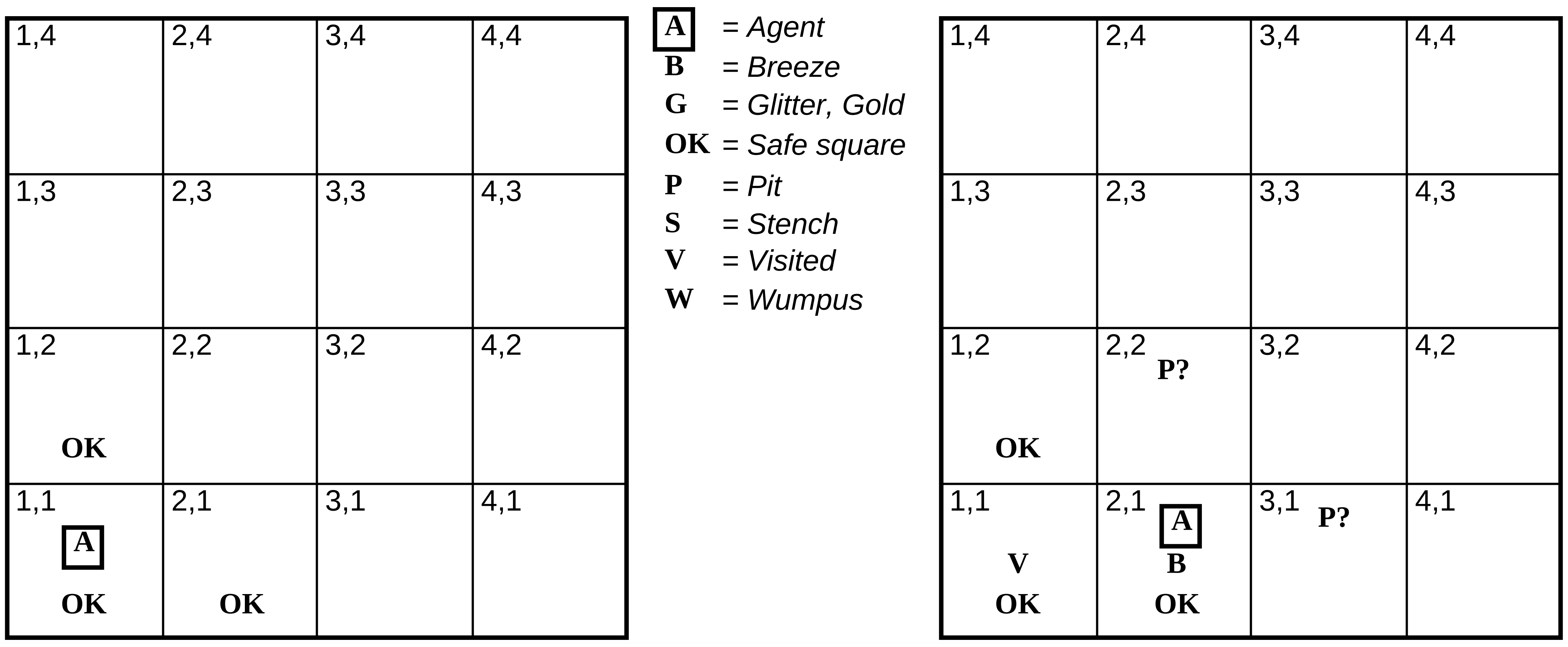
- $\mathbf{see} : \mathbf{S} \longrightarrow \mathbf{P}$,
- $\mathbf{next} : \mathbf{D} \times \mathbf{P} \longrightarrow \mathbf{D}$,

```
function action(Δ : D) : A
begin
    for each a ∈ A do
        if Δ ⊢_ρ Do(a) then
            return a
        end-if
    end-for
    for each a ∈ A do
        if Δ ⊬_ρ ¬Do(a) then
            return a
        end-if
    end-for
    return null
end function action
```

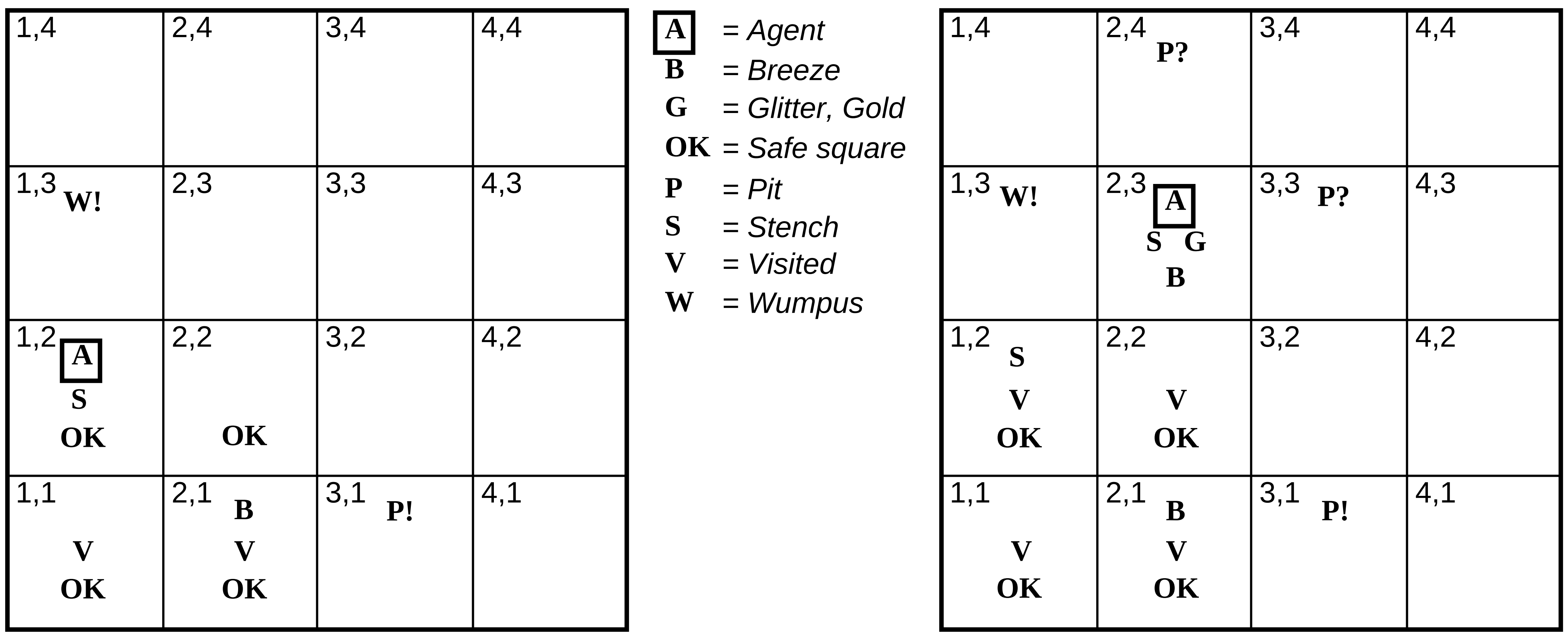
3.1 Sentential Logic SL

The Wumpus-World in SL




1,4 2,4 3,4 4,4
1,3 2,3 3,3 4,3
1,2 2,2 3,2 4,2
OK
1,1 2,1 3,1 4,1
A
OK OK
A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus
1,4 2,4 3,4 4,4
1,3 2,3 3,3 4,3
1,2 2,2 P? 3,2 4,2
OK
1,1 2,1 A 3,1 P? 4,1
V B
OK OK

(a) (b)


1,4
2,4
3,4
4,4
1,3 W!
2,3
3,3
4,3
1,2 A S OK
2,2 OK
3,2
4,2
1,1 V OK
2,1 B V OK
3,1 P!
4,1
A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus
1,4
2,4 P?
3,4
4,4
1,3 W!
2,3 A S G B
3,3 P?
4,3
1,2 S V OK
2,2 V OK
3,2
4,2
1,1 V OK
2,1 B V OK
3,1 P!
4,1

(a) (b)

Defining the language:

$S_{i,j}$	stinks
$B_{i,j}$	is cold
$Pit_{i,j}$	is a pit
$Gl_{i,j}$	glitters
$W_{i,j}$	contains Wumpus

General Knowledge:

$$\neg S_{1,1} \longrightarrow (\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1})$$

$$\neg S_{2,1} \longrightarrow (\neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1})$$

$$\neg S_{1,2} \longrightarrow (\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3})$$

$$S_{1,2} \longrightarrow (W_{1,3} \wedge W_{1,2} \wedge W_{2,2} \wedge W_{1,1})$$

Knowledge after the 3rd move:

$$\neg S_{1,1} \wedge \neg S_{2,1} \wedge S_{1,2} \wedge \neg B_{1,1} \wedge \neg B_{2,1} \wedge \neg B_{1,2}$$

Can we deduce that the Wumpus is in $(1,3)$?

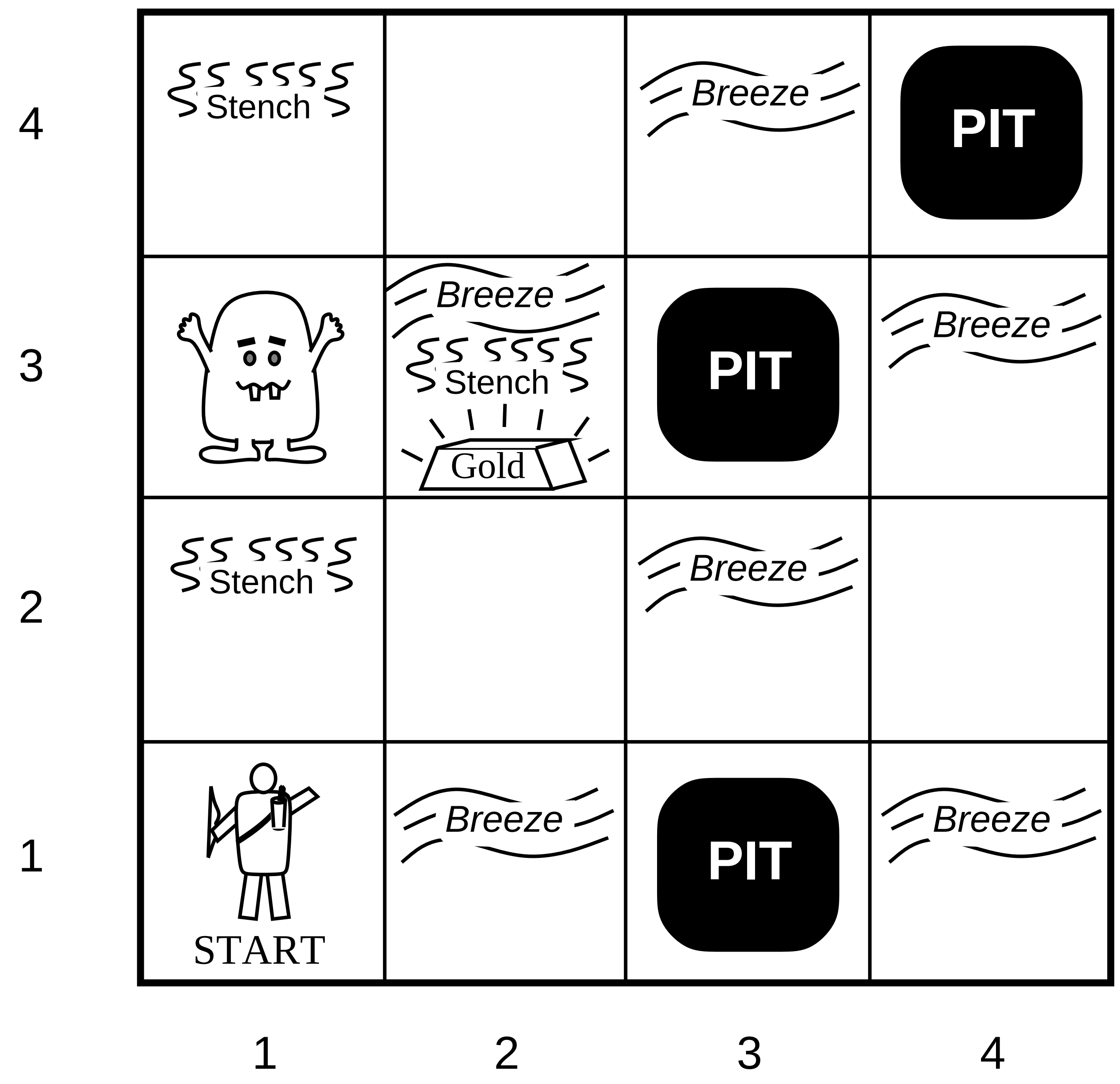
Yes, with any reasonable calculus.

But we want much more: for a given situation find the **best suited action**.

That does not work in SL. We can only check for each action, whether it should be executed or not. Even for this we need additional axioms:

Additional axioms:

$$A_{1,1} \land East \land W_{2,1} \longrightarrow \neg Forward$$
$$A_{1,1} \land East \land Grube_{2,1} \longrightarrow \neg Forward$$
$$A_{i,j} \land Gl_{i,j} \longrightarrow Take_{Gold}$$


Stench
Breeze
PIT
Breeze
Stench
Gold
PIT
Breeze
Stench
Breeze
START
Breeze
PIT
Breeze
1
2
3
4

3.2 The Situation Calculus

How can we represent a dynamic, changing world?
How can we formalize the wumpus world in it?

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*

Idea: To describe actions and their effects consistently, we represent the world as a sequence of situations (snapshots of the world).

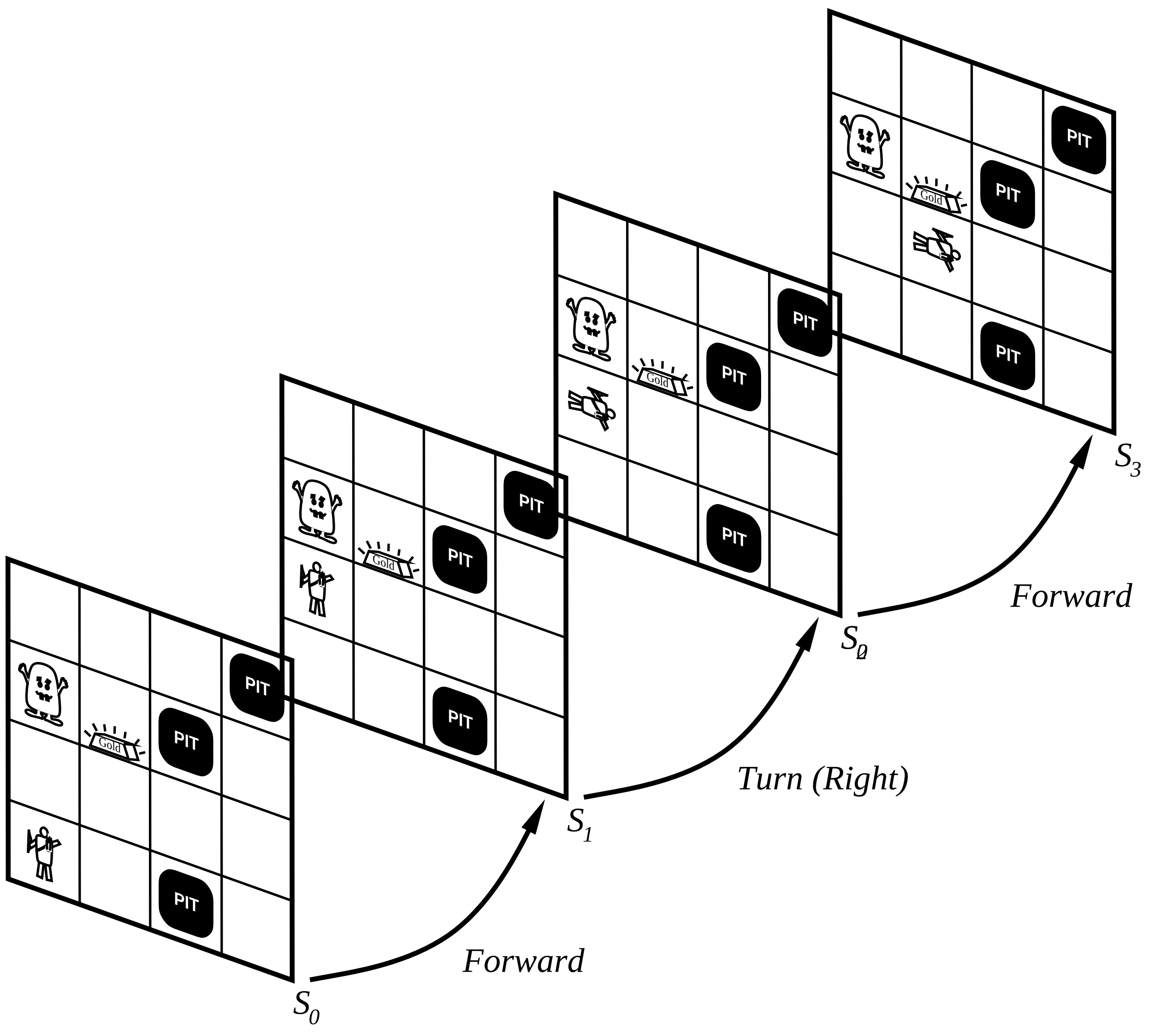
To do this, we have to extend each predicate by an additional argument (representing the situation we are in).

We use a function symbol

result(action, situation)

which represents a term for the situation wich occurs when in situation **situation** the action **action** is executed (**History**).

actions: **turn_right**, **turn_left**, **forward**, **shoot**, **grab**, **release**, **climb**.


PIT
S_0
Forward
S_1
Turn (Right)
S_2
Forward
S_3

We also need a memory: this is a ternary predicate

$$\mathbf{At}(person, location, \mathbf{situation})$$

where *person* can be *wumpus* or *agent* and *location* stands for the current location, coded as a pair $[i, j]$.

Important axioms are the

“Successor-state Axioms”.

They describe the effects of actions to the situations. Their general form is

true afterwards $\Longleftrightarrow$ an **action** made it true
or it is already true and
no action made it false

Axioms for $\mathbf{At}(p,l,\mathbf{s})$:

$$\begin{aligned}
\mathbf{At}(p,l,\mathbf{result}(\mathbf{a},\mathbf{s})) &\leftrightarrow ((l = location_ahead(p,\mathbf{s}) \wedge \neg Wall(l) \wedge \mathbf{a} = \mathbf{forward}) \\
&\quad \vee (\mathbf{At}(p,l,\mathbf{s}) \wedge \neg \mathbf{a} = \mathbf{forward}))
\end{aligned}$$

$$\begin{aligned}
\mathbf{At}(p,l,\mathbf{s}) &\rightarrow location_ahead(p,\mathbf{s}) = location_toward(l, orient.(p,\mathbf{s})) \\
Wall([x,y]) &\leftrightarrow (x = 0 \vee x = 5 \vee y = 0 \vee y = 5)
\end{aligned}$$

$$
\begin{aligned}
&location\_toward([x,y],0) = [x+1,y]\\
&location\_toward([x,y],90) = [x,y+1]\\
&location\_toward([x,y],180) = [x-1,y]\\
&location\_toward([x,y],270) = [x,y-1]
\end{aligned}
$$

$$orient.(agent, s_0) = 90$$

$$
\begin{aligned}
orient.(p, \mathbf{result}(\mathbf{a}, \mathbf{s})) = d \quad \leftrightarrow \quad & \big((\mathbf{a} = \mathbf{turn\_right} \wedge d = mod(orient.(p, \mathbf{s}) - 90, 360))\\
& \vee(\mathbf{a} = \mathbf{turn\_left} \wedge d = mod(orient.(p, \mathbf{s}) + 90, 360))\\
& \vee(orient.(p, \mathbf{s}) = d \wedge \neg(\mathbf{a} = \mathbf{turn\_right} \vee \mathbf{a} = \mathbf{turn\_left}))
\end{aligned}
$$

Here $mod(x,y)$ is a built-in “modulo”-function: each x is reduced to a unique value between 0 and y.

Axioms for observations, extension by definition:

$$\begin{array}{lcl}
Percept([stench,b,g,u,c],\mathbf{s}) & \rightarrow & Stench(\mathbf{s}) \\
Percept([a,breeze,g,u,c],\mathbf{s}) & \rightarrow & Breeze(\mathbf{s}) \\
Percept([a,b,glitter,u,c],\mathbf{s}) & \rightarrow & \mathbf{At}_gold(\mathbf{s}) \\
Percept([a,b,g,bump,c],\mathbf{s}) & \rightarrow & \mathbf{At}_wall(\mathbf{s}) \\
Percept([a,b,g,u,scream],\mathbf{s}) & \rightarrow & Wumpus_dead(\mathbf{s})
\end{array}$$

$$\begin{array}{lcl}
\mathbf{At}(agent,l,\mathbf{s}) \wedge Breeze(\mathbf{s}) & \rightarrow & Breezy(l) \\
\mathbf{At}(agent,l,\mathbf{s}) \wedge Stench(\mathbf{s}) & \rightarrow & Smelly(l)
\end{array}$$

$$
\begin{aligned}
Adjacent(l_1, l_2) &\leftrightarrow \exists d\ l_1 = location\_toward(l_2, d) \\
Smelly(l_1) &\rightarrow \exists l_2\ \mathbf{At}(wumpus, l_2, \mathbf{s}) \wedge (l_2 = l_1 \vee Adjacent(l_1, l_2))
\end{aligned}
$$

$$
\begin{aligned}
Percept([none, none, g, u, c], \mathbf{s}) \wedge \mathbf{At}(agent, x, \mathbf{s}) \wedge Adjacent(x, y) &\rightarrow OK(y) \\
(\neg \mathbf{At}(wumpus, x, t) \wedge \neg Pit(x)) &\rightarrow OK(y)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{At}(wumpus, l_1, \mathbf{s}) \wedge Adjacent(l_1, l_2) &\rightarrow Smelly(l_2) \\
\mathbf{At}(Pit, l_1, \mathbf{s}) \wedge Adjacent(l_1, l_2) &\rightarrow Breezy(l_2)
\end{aligned}
$$

Axioms to describe actions:

$$
\begin{array}{lll}
Holding(gold, \mathbf{result}(\mathbf{grab}, \mathbf{s})) & \leftrightarrow & (\mathbf{At\_}gold(\mathbf{s}) \vee Holding(gold, \mathbf{s})) \\
Holding(gold, \mathbf{result}(\mathbf{release}, \mathbf{s})) & \leftrightarrow & \Box \\
Holding(gold, \mathbf{result}(\mathbf{turn\_right}, \mathbf{s})) & \leftrightarrow & Holding(gold, \mathbf{s}) \\
Holding(gold, \mathbf{result}(\mathbf{turn\_left}, \mathbf{s})) & \leftrightarrow & Holding(gold, \mathbf{s}) \\
Holding(gold, \mathbf{result}(\mathbf{forward}, \mathbf{s})) & \leftrightarrow & Holding(gold, \mathbf{s}) \\
Holding(gold, \mathbf{result}(\mathbf{climb}, \mathbf{s})) & \leftrightarrow & Holding(gold, \mathbf{s})
\end{array}
$$

All effects have to be carefully described.

Axioms to describe preferences between actions:

$$Great(\mathbf{a},\mathbf{s}) \rightarrow Action(\mathbf{a},\mathbf{s})$$
$$(Good(\mathbf{a},\mathbf{s}) \wedge \neg\exists \mathbf{b}\, Great(\mathbf{b},\mathbf{s})) \rightarrow Action(\mathbf{a},\mathbf{s})$$
$$(Medium(\mathbf{a},\mathbf{s}) \wedge \neg\exists \mathbf{b}\, (Great(\mathbf{b},\mathbf{s}) \vee Good(\mathbf{b},\mathbf{s}))) \rightarrow Action(\mathbf{a},\mathbf{s})$$

$$\mathbf{At}(agent,[1,1],\mathbf{s}) \wedge Holding(gold,\mathbf{s}) \rightarrow Great(\mathbf{climb},\mathbf{s})$$
$$\mathbf{At\_}gold(\mathbf{s}) \wedge \neg Holding(gold,\mathbf{s}) \rightarrow Great(\mathbf{grab},\mathbf{s})$$
$$\mathbf{At}(agent,l,\mathbf{s}) \wedge \neg Visited(location\_ahead(agent,\mathbf{s})) \wedge$$
$$\wedge OK(location\_ahead(agent,\mathbf{s})) \rightarrow Good(\mathbf{forward},\mathbf{s})$$

$$Visited(l) \leftrightarrow \exists s\, \mathbf{At}(agent,l,\mathbf{s})$$

We do not just want to find the gold, we also want to come back alive! Therefore one needs axioms like $Holding(gold,\mathbf{s}) \rightarrow \mathbf{Go\_back}(\mathbf{s})$.

3.3 Problems

There are three very important problems in axiomatizing a dynamically changing world:

Frame problem: **actions usually change very little. But one needs a huge number of actions to describe invariant properties.**

It would be much better to **axiomatize only what does not persist** and assume that **nothing else changes**.

Qualification problem: **We need to enumerate all conditions under which an action is successful.** E.g.

$$\forall x \quad (\mathbf{Bird}(x) \quad \wedge\neg\mathbf{Penguin}(x)\wedge\neg\mathbf{Dead}(x)\wedge \\ \wedge\neg\mathbf{Ostrich}(x)\wedge\neg\mathbf{Broken\_wings}(x)\wedge \\ \wedge\ldots \\ \longrightarrow \mathbf{Flies}(x)$$

It would be much better to simply assume **birds normally fly**.

Ramification problem: **How to deal with implicit consequences of actions?** E.g. $\mathbf{grab}(gold)$. $gold$ could be radioactive after this action is executed. Then the action $\mathbf{grab}(gold)$ is not optimal.

Programming versus Knowledge Engineering.

Programming	**Knowledge Engineering**
Choose programming language.	Choose **Logic**.
Write program.	Define **Knowledge Base**.
Write compiler.	Implement **Calculus**.
Execute program.	Deduce new **facts**.

3.4 A Solution to the Frame Problem?

Successor State Axioms

Where do the successor state axioms come from?

- We have to ask: **Which fluents stay invariant?**

We distinguish between two sorts of fluents:

relational fluent:

$$\neg \mathbf{broken}(x,\mathbf{s}) \wedge (x \neq y \vee \neg fragile(x,z)) \longrightarrow \neg \mathbf{broken}(x,\mathbf{result}(\mathbf{drop}(r,y),\mathbf{s}))$$

functional fluent:

$$\mathbf{color}(x,\mathbf{s}) = \mathbf{c} \longrightarrow \mathbf{color}(x,\mathbf{result}(\mathbf{drop}(r,y),\mathbf{s})) = \mathbf{c}$$

How many of such axioms do we need?

We need exactly

$$\mathbf{2} \times \mathbf{\#actions} \times \mathbf{\#fluents}$$

Suppose we are given axioms of the form

$$\begin{aligned} \ldots &\longrightarrow \mathbf{fluent}(x, \mathbf{result}(\mathbf{action}, \mathbf{s})) \\ \ldots &\longrightarrow \neg\mathbf{fluent}(x, \mathbf{result}(\mathbf{action}, \mathbf{s})), \end{aligned}$$

how can we compute the successor state axioms **automatically**?

Note, that the above set assumes implicitly that all actions can be applied: this is an overly optimistic assumption according to the Qualification Problem.

Qualification Problem Revisited

We assume a predicate **Poss** to describe the possibility to apply an action.

$$\mathbf{Poss}(\mathbf{pickup}(r,x),\mathbf{s}) \rightarrow \forall z\,(\neg holding(r,z,\mathbf{s})\,).$$

But $\rightarrow$ is too weak. Can we replace it by $\leftrightarrow$?

What about

$$\mathbf{Poss}(\mathbf{pickup}(r,x),\mathbf{s}) \rightarrow \forall z\,(\neg holding(r,z,\mathbf{s}) \wedge \neg heavy(x) \wedge nextto(r,x,\mathbf{s}))\,.$$

We suppose we are given a list of axioms of the form

$$\mathbf{Poss}(\mathbf{action}(\underline{x}),\mathbf{s}) \longleftrightarrow \phi_{\mathbf{action}}(\underline{x},\mathbf{s})$$

where $\phi_{\mathbf{action}}(\underline{x},\mathbf{s})$ does not contain any **result**-terms.

$$(1): \quad fragile(x,\mathbf{s}) \longrightarrow broken(x,\mathbf{result}(\mathbf{drop}(r,x),\mathbf{s}))$$
$$(1'): \quad nextto(b,x,\mathbf{s}) \longrightarrow broken(x,\mathbf{result}(\mathbf{explode}(b),\mathbf{s}))$$
$$(2): \quad \longrightarrow \neg broken(x,\mathbf{result}(\mathbf{repair}(r,x),\mathbf{s}))$$

We assume these are **all** possibilities for *broken*, $\neg$*broken*. Then $(1),(1')$ are equivalent to

$$\exists r\,(a = \mathbf{drop}(r,x) \wedge fragile(x,\mathbf{s}))\vee$$
$$\exists b\,(a = \mathbf{explode}(b) \wedge nextto(b,x,\mathbf{s}))$$
$$\longrightarrow$$
$$broken(x,\mathbf{result}(\mathbf{a},\mathbf{s})).$$

(2) is equivalent to

$$\exists r\,\mathbf{a} = \mathbf{repair}(r,x) \longrightarrow \neg broken(x,\mathbf{result}(\mathbf{a},\mathbf{s}))$$

Under which conditions could $\neg broken(x, \mathbf{s})$ and $broken(x, \mathbf{result}(\mathbf{a}, \mathbf{s}))$ be both true?

$$(1''): \quad \neg broken(x, \mathbf{s}) \wedge broken(x, \mathbf{result}(\mathbf{a}, \mathbf{s})) \quad \longrightarrow \quad \exists r\, (\mathbf{a} = \mathbf{drop}(r, x) \wedge fragile(x, \mathbf{s}) \vee \exists b\, (\mathbf{a} = \mathbf{explode}(b) \wedge nextto(b, x, \mathbf{s}))$$

$$(2'): \quad broken(x, \mathbf{s}) \wedge \neg broken(x, \mathbf{result}(\mathbf{a}, \mathbf{s})) \quad \longrightarrow \quad \exists r\, \mathbf{a} = \mathbf{repair}(r, x)$$

$(1),(1'),(2),(1''),(2')$ are equivalent to the **successor state axiom**

$$\begin{aligned} broken(x, \mathbf{result}(\mathbf{a}, \mathbf{s})) \longleftrightarrow\ & \exists r\,(\mathbf{a} = \mathbf{drop}(r,x) \wedge fragile(x,\mathbf{s})) \vee \\ & \exists b\,(\mathbf{b} = \mathbf{explode}(b) \wedge nextto(b,x,\mathbf{s})) \vee \\ & broken(x,\mathbf{s}) \wedge \neg\exists r\, \mathbf{a} = \mathbf{repair}(r,x) \end{aligned}$$

This can be generalized, also for functional fluents!

Thus the $\mathbf{2} \times$ **#actions** $\times$ **#fluents** many axioms can be rewritten into only

#fluents

many axioms ($\mathbf{2} \times$ **#fluents** if we count each equivalence twice). But we also need the **Poss** axioms: another **#actions** many.

Altogether, the $\mathbf{2} \times$ **#actions** $\times$ **#fluents** are compiled into (modulo a constant factor)

#actions+**#fluents**.

Some people call this a solution to the frame problem.

References

Arisha, K., F. Ozcan, R. Ross, V. S. Subrahmanian, T. Eiter, and S. Kraus (1999, March/April). IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems 14*, 64–72.

Bratman, M., D. Israel, and M. Pollack (1988). Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence 4*(4), 349–355.

Dix, J., S. Kraus, and V. Subrahmanian (2001). Temporal agent reasoning. *Artificial Intelligence to appear*.

Dix, J., M. Nanni, and V. S. Subrahmanian (2000). Probabilistic agent reasoning. *Transactions of Computational Logic 1*(2).

Dix, J., V. S. Subrahmanian, and G. Pick (2000). Meta Agent Programs. *Journal of Logic Programming 46*(1-2), 1–60.

Eiter, T., V. Subrahmanian, and G. Pick (1999). Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence 108*(1-2), 179–255.

Eiter, T., V. Subrahmanian, and T. J. Rogers (2000). Heterogeneous Active Agents, III: Polynomially Implementable Agents. *Artificial Intelligence 117*(1), 107–167.

Eiter, T. and V. S. Subrahmanian (1999). Heterogeneous Active Agents, II: Algorithms and Complexity. *Artificial Intelligence 108*(1-2), 257–307.

Georgeff, M. and A. Lansky (1987). Reactive Reasoning and Planning. In *Proceedings of the Conference of the American Association of Artificial Intelligence*, Seattle, WA, pp. 677–682.

Rao, A. S. (1995). Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics. In M. Wooldridge, J. Müller, and M. Tambe (Eds.), *Intelligent Agents II – Proceedings of the 1995 Workshop on Agent Theories, Architectures and Languages (ATAL-95)*, Volume 890 of *LNAI*, pp. 1–39. Berlin, Germany: Springer-Verlag.

Rao, A. S. and M. Georgeff (1991). Modeling Rational Agents within a BDI-Architecture. In J. F. Allen, R. Fikes, and E. Sandewall (Eds.), *Proceedings of the International Conference on Knowledge Representation and Reasoning*, Cambridge, MA, pp. 473–484. Morgan Kaufmann.

Rao, A. S. and M. Georgeff (1995, June). Formal models and decision procedures for multi-agent systems. Technical Report 61, Australian Artificial Intelligence Institute, Melbourne.

Subrahmanian, V., P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross (2000). *Heterogenous Active Agents*. MIT-Press.

Weiss, G. (Ed.) (1999). *Multiagent Systems*. MIT-Press.