

Situation Calculus



Prof. Mauro Gaspari
Dipartimento di Informatica Scienze e Ingegneria
(DISI)

mauro.gaspari@unibo.it



Representing changes



- The predicate calculus works well as a representation language for static domains, where nothing ever changes, and whatever is true, stays so forever.
- Unfortunately, the real world is not like this.
- Example:
 - The formula $\text{At}(\text{John}, \text{School})$ correctly describes the current state of the morning of some weekday.
 - However John will go home eventually and the system might be able to derive a new fact: $\text{At}(\text{John}, \text{Home})$.
 - This can be a contradiction, which problem for a FOL based system.



Temporal Logics



- To solve the problem of representation of the changes, a number KR formalisms were created, for example **temporal logics**.
- Ordinary facts expressed in these logics occur at specific points in time. However, the time, its properties, and special inference rules concerning its passage, are built into the theory (instead of being represented explicitly, along with other properties of the world).
- Moreover, temporal logics are more complex, indeed they belongs from the modal logics family and they require modal operators.
- An alternative to temporal logics is a direct recording of time moments in the representation language. An example of such an approach is the **situation calculus**.



Situation Calculus



- **Reasoning about actions** is a fundamental issue for KB systems that represents an agent (wumpus, monkey).
- The approach of the **situation calculus** uses the concept of **situations** to denote states resulting from executing actions.
- The situation calculus uses FOL syntax and establishes a set of conventions that should be used to represent an agent that performs actions.
- We assume that the environment contains only one agent. To represent more agents an additional argument should be inserted to represent which agent is doing the action.

Situation Calculus



- **Situations:** are logical terms consisting of the initial situation (called S_0), and all situations that are generated by applying an action to a situation. The function $\text{Result}(a, s)$ individuates the situation that results from the execution of action a in the situation s . Situations are associated to sequences of actions, and are thus different from states, ie. an agent may be in some state through different situations, 
- **Fluents:** functions and relations which can vary from one situation to the next are called fluents; by convention their last argument is the situation argument. 
- **Atemporal or eternal:** predicates and functions that does not change over time. For example: $\text{Gold}(\text{gold1})$

Sequences of Actions



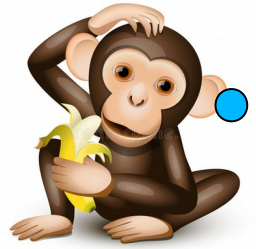
- In addition to single actions, it is useful to reason about action sequences:
- Executing the empty sequence leaves the situation unchanged:

$$\text{Result}([],s) = s$$

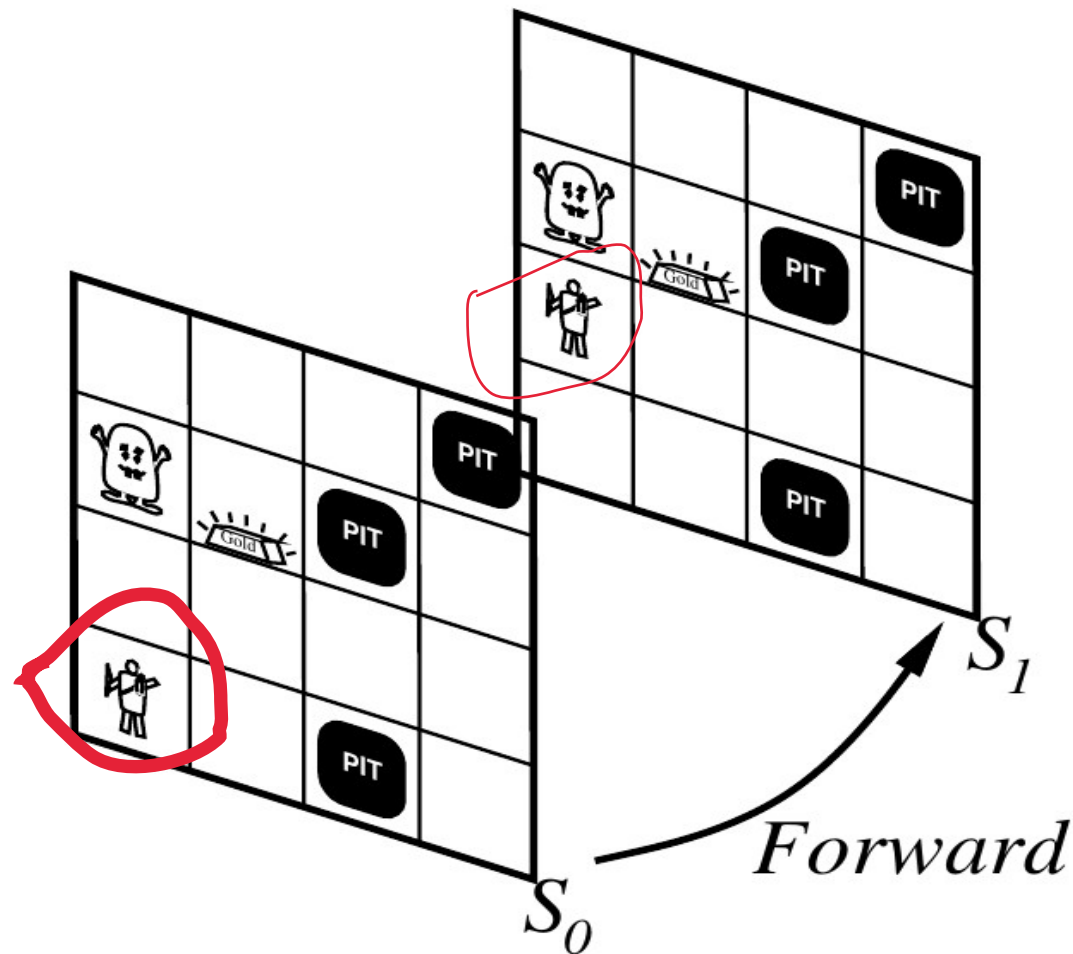
- Executing a non empty sequence means executing the first action and then executing the rest:

$$\text{Result}([a|seq],s) = \text{Result}(seq, \text{Result}(a,s))$$

Example

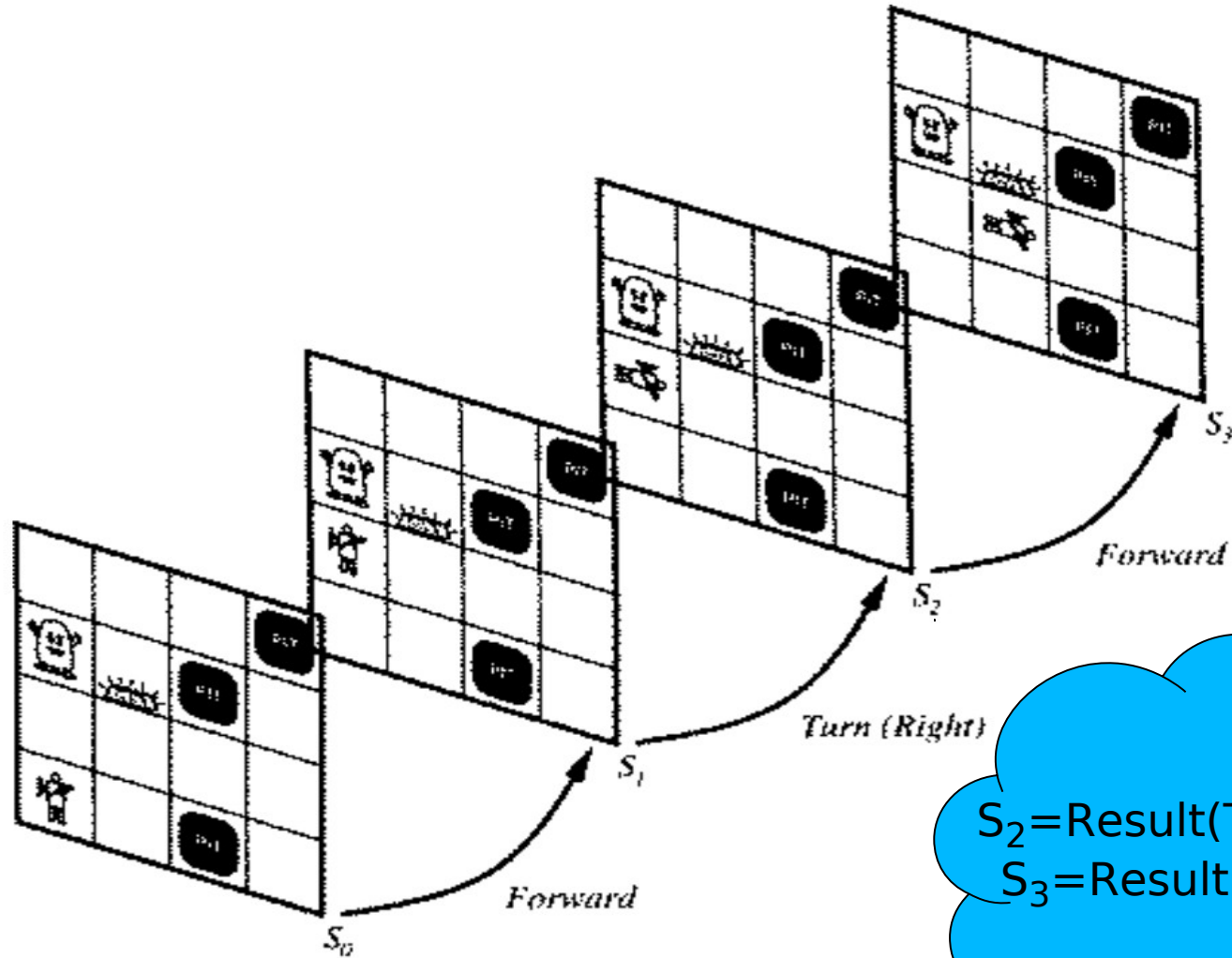


- The wumpus world:



Result(Forward, S_0)

Example



$S_2 = \text{Result}(\text{Turn}(\text{Right}), S_1)$
 $S_3 = \text{Result}(\text{Forward}, S_2)$

Tasks of an Agent



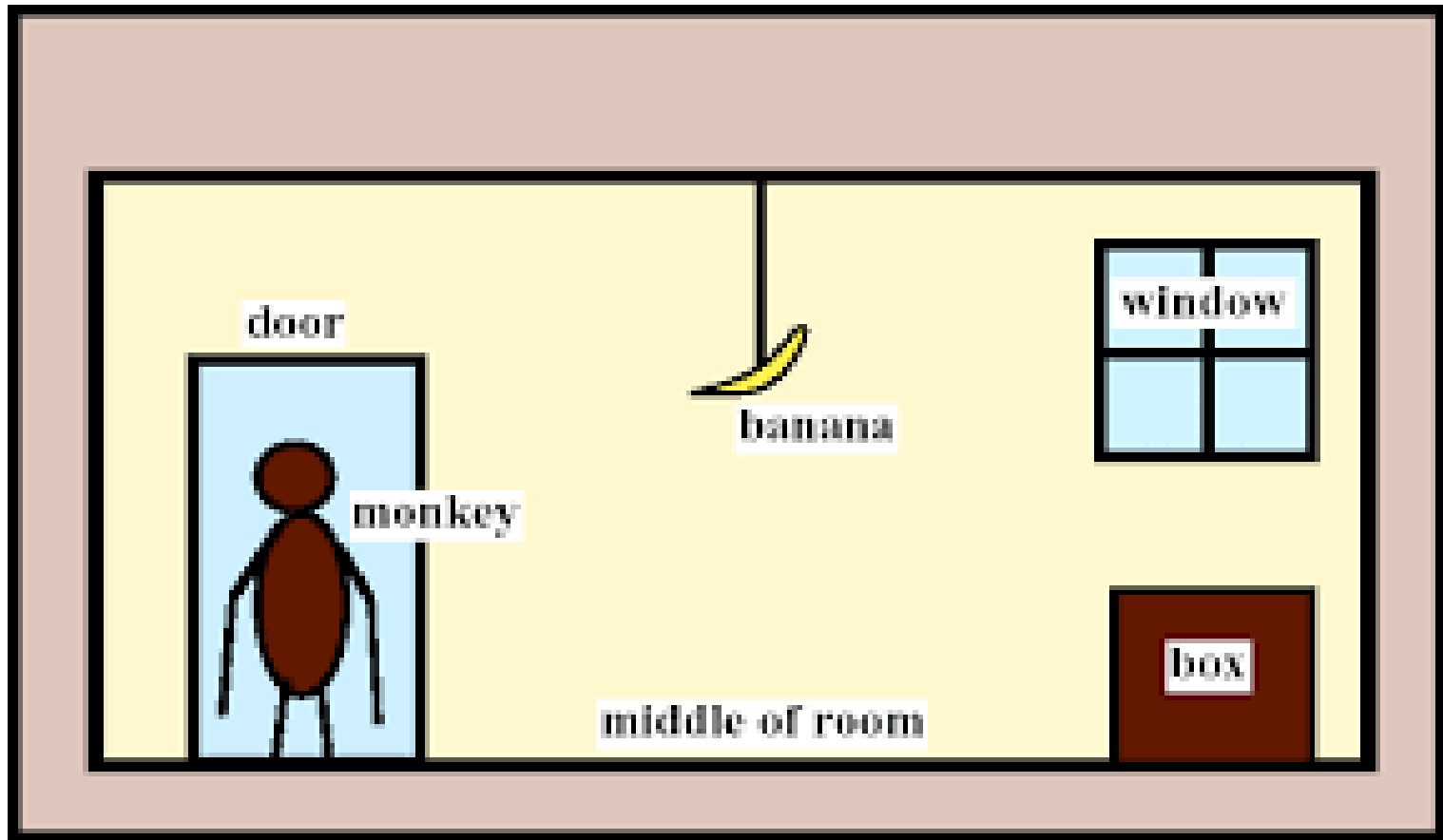
- **Projection task:** an agent should be able to deduce the outcome of a given sequence of actions.
- **Planning task:** an agent using a constructive inference algorithms should be able to find a sequence to achieve a desired effect.

Axioms



- In the simplest version of situation calculus each action is described by two actions:
 - **Possibility axiom:** it express when it is possible to execute the action.
 - **Effect axiom:** expresses what happens when a possible action is executed.
- We will use the notation **Poss(a,s)** to express that it is possible to execute action **a** in situation **S**.
- These axioms have the form:
POSSIBILITY AXIOM: $\text{Preconditions} \rightarrow \text{Poss}(a,s)$
EFFECT AXIOM: $\text{Poss}(a,s) \rightarrow \text{Changes executing } a$.

Monkey and Banana



Fluent predicates



- The **fluent predicates** we use for representing the monkey and banana problem are:
 - $\text{At}(\text{Agent}, \text{Position}, s) \implies$ The monkey is the Agent, possible positions are door, middle, windows.
 - $\text{HasBanana}(\text{Agent}, s) \implies$ the agent has got the banana.
- **Atemporal predicates:**
 - $\text{Walk}(x, y) \implies$ it includes all the possible places x and y which it is possible to move through. It does not change.



Representing Actions



- $\text{Go}(x,y) \implies$ The agent moves from x to y .
- $\text{Push}(x,y) \implies$ The agent pushes box from x to y .
- $\text{Climb} \implies$ the agent goes on the box.
- $\text{Grab} \implies$ the agent grab the banana.

Possibility Axioms



- We assume variables as universally quantified.
- $\text{At}(\text{Agent}, x, s) \wedge \text{walk}(x, y) \rightarrow \text{Poss}(\text{Go}(x, y), s)$
- $\text{At}(\text{Agent}, x, s) \wedge \text{At}(\text{box}, x, s) \wedge \text{walk}(x, y) \rightarrow \text{Poss}(\text{Push}(x, y), s)$
- $\text{At}(\text{Agent}, x, s) \wedge \text{At}(\text{box}, x, s) \rightarrow \text{Poss}(\text{Climb}, s)$
- $\text{At}(\text{Agent}, \text{onbox}, s) \wedge \text{At}(\text{box}, \text{middle}, s) \wedge \text{At}(\text{banana}, \text{middle}, s) \rightarrow \text{Poss}(\text{Grab}, s)$

Effect Axioms



- We assume that variables are universally quantified.
- $\text{Poss}(\text{Go}(x,y),s) \rightarrow \text{At}(\text{Agent},y,\text{Result}(\text{Go}(x,y),s))$
- $\text{Poss}(\text{Push}(x,y),s) \rightarrow$
 $\text{At}(\text{Agent},y,\text{Result}(\text{Go}(x,y),s)) \wedge$
 $\text{At}(\text{box},y,\text{Result}(\text{Go}(x,y),s))$
- $\text{Poss}(\text{Climb},s) \rightarrow \text{At}(\text{Agent},\text{onbox},\text{Result}(\text{climb},s))$
- $\text{Poss}(\text{Grab},s) \rightarrow \text{HasBanana}(\text{Agent},\text{Result}(\text{grab},s)) \wedge$
 $\neg \text{At}(\text{banana},\text{middle},s)$

Initial State and Query



- The initial KB include the following predicates:

At(monkey,door, s_0)

At(box>window, s_0)

At(banana,middle, s_0)

\neg HasBanana(monkey, s_0)

walk(door,middle)

walk(door>window)

walk>window,middle)

- A possible query could be:

\exists seq HasBanana(monkey,Result(seq, s_0))

Frame Problem



- One problem with the situation calculus is to specify what does not change when a given action is executed.
- For example the predicate $\text{At}(\text{banana}, \text{middle}, s_0)$ holds in s_0 but we do not know if it holds in s .
- The problem is that the effect axioms says what changes but do not say what stays the same.
- Representing all the things that stay the same is called the **Frame Problem**.
- Note that in real world almost everything stays the same almost all the time. Each action affect only a tiny fraction of all fluents.

Frame Problem Solutions



- One approach is to write explicit frame axioms that specify what stays the same for each action:

$$\text{At}(o,x,s) \wedge o \neq \text{monkey} \wedge \neg \text{HasBanana}(o,s) \\ \rightarrow \text{At}(o,x,\text{result}(\text{Go}(x,y),s))$$

- If there are F fluent predicates and A actions then we need $A * F$ frame axioms.
- These axioms are hard to state in a general way and they significantly complicate the representation.
- This is known as the **representational frame problem**.

Representational Frame Problem



- The solution to the representational frame problem involves just a slight change of viewpoint and how to write the axioms.
- Instead of writing out the effects of each action, we consider how each fluent predicate evolves over time.
- These axioms are called **successor-state-axioms**. They have the following form:

SUCCESSOR-STATE-AXIOM:

Action is possible \rightarrow

(Fluent is true in result state \leftrightarrow effect of action made true \vee

It was true before and action left it alone)





Observation



- Only possible actions are considered.
- The definition that follows is stronger: it uses \leftrightarrow : this means that the fluent will be True if and only if the right-hand side holds.
- Intuitively this means that these actions specify the truth value of each fluent in the next state as a function of the action and the truth values in the current state.
- In this way the next state is completely specified and additional frame axioms are not needed.
- Successor-state-axioms solve the representation frame problem because the total size of the axioms is about $A * E$.

Example



- The first effect axiom which concern the $Go(x,y)$ action can be transformed in the following successor-state-axiom :

$Poss(a,s) \rightarrow$

$$At(Agent,y,Result(a,s)) \leftrightarrow a=Go(x,y) \vee \\ (At(Agent,y,s) \wedge a \neq Go(x,y))$$

Unique action axioms



- Given that successor-state-axioms need inequality we must add specific axioms to effectively derive such facts.
- To this purpose we can use the following **unique action axioms**:
- For each pair of action symbols A_i and A_j we must state the (seemingly obvious) axiom:

$$A_i \neq A_j$$

$$A_i(x_1, \dots, x_n) \neq A_j(y_1, \dots, y_n)$$

- and for actions with parameters we must also state:

$$A(x_1, \dots, x_n) = A(y_1, \dots, y_n) \leftrightarrow x_1 = y_1 \wedge \dots \wedge x_n = y_n$$

Lack of information



- The logic-based methods presented so far assumed that all information necessary to carry out logical reasoning is available to the agent. Unfortunately, this is not a realistic assumption.
- One problem is that of **incomplete information**. An Agent may not have full information about a problem to draw categorical conclusions.
- It should be able to reason with partial information, such as:
 - typical facts,
 - possible facts,
 - probable facts,
 - exceptions to the generally valid facts.

Limitations of FOL



- Having such information is often crucial for making the right decisions.
- Unfortunately, the classical predicate logic cannot make any use of them.
- Another problem is the uncertainty of information. An agent may have data from a variety of not fully reliable sources. In the absence of certain information, those unreliable data should be used. She should reason using the best available data, and estimate the reliability of any conclusions obtained this way.
- Again, FOL (classical logic) does not provide such tools.

Common sense reasoning



- Consider what information a human knows for sure, making decisions in everyday life.
- For example:
 - Getting up in the morning, his intention is to go to work. But what if there is a large-scale failure of public transportation? She should, in fact, get up much earlier, and first check whether the buses are running.
 - The day before, she bought products to make breakfast. But can she be sure that her breakfast salad is still in the refrigerator, or if it did not spoil, or perhaps if someone has not sneaked to steal it, etc.

Common sense reasoning



- A logically reasoning agent needs 100% certain information to conduct its actions, and sooner or later it will be paralyzed by the perfect correctness of her inference system.
- In the real world it will never be able to undertake any action, without full information about the surrounding world.
- However, people perform quite well in a world full of uncertain and incomplete information, defaults facts, and exceptions. How do they do it?
- We must conclude that, in their reasoning, the humans use a slightly different logic than the rigorous mathematical logic: a logic of common sense reasoning.

References and Tools



- Cognitive Robotics University of Toronto:
<http://www.cs.toronto.edu/cogrobo/main/systems/index.html>
- Haskell implementation of the situation calculus and action language Golog:
<https://github.com/schwering/golog>
- Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, Richard B. Scherl, GOLOG: A logic programming language for dynamic domains, The Journal of Logic Programming, Volume 31, Issues 1–3, 1997, Pages 59-83.