Artificial Intelligence 1

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Knowledge Representation

How to effectively construct a **knowledge base**?
How should axioms look like?

- Representing **objects**
  - objects, categories, and ontologies
- Representing **time** and **actions**
  - situation calculus
  - frame problem

Let us notice that

- agents **manipulate** with real **objects**
- but **reasoning** is done at the level of **categories**
- An agent uses observations to find properties of objects that are used to assign objects to categories. Reasoning on category then reveals useful information about the object itself.

Category

= a set of its members

= a complex object with relations

- MemberOf
- SubsetOf

How to represent a category in FOL?

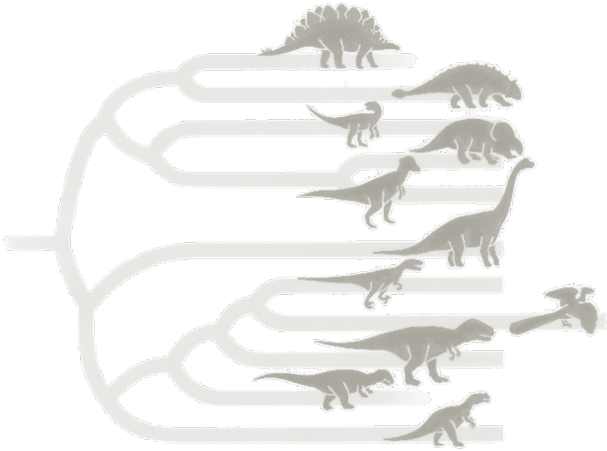
- an object is a **member** of a category
  - MemberOf(BB_{12},Basketballs)
- a category is **subset** of another category
  - SubsetOf(Basketballs,Balls)
- all members of the category have some **property**
  - $\forall x$ (MemberOf(x,Basketballs) $\Rightarrow$ Round(x))
- all members of the category can be **recognized** using common properties
  - $\forall x$ (Orange(x) $\wedge$ Round(x) $\wedge$ Diameter(x)=9.5in $\wedge$ MemberOf(x,Balls) $\Rightarrow$ MemberOf(x,BasketBalls))
- category may also have some property
  - MemberOf(Dogs,DomesticatedSpecies)

Categories organize and simplify knowledge base by using **inheritance of properties**.

- properties are defined for a category, but they are inherited to all members of the category
- food is eatable, fruits are food, apples are fruits, and hence apples are eatable

Subclasses organize categories to a **taxonomy**

- a hierarchical structure that is used to categorize objects
- originally proposed for classifying living organisms (alpha taxonomy)
- categories for all knowledge
  - Used in libraries
  - Dewey Decimal Classification
  - 330.94 European economy



So far we modelled a static world only.

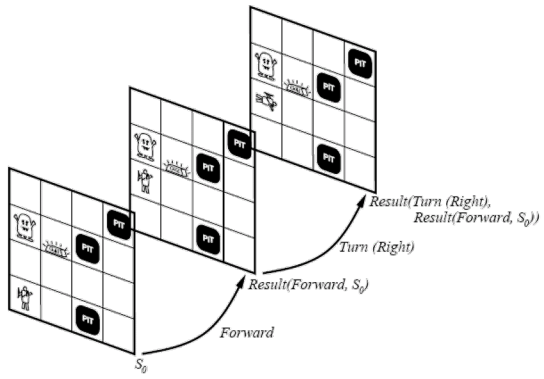
How to reason about actions and their effects in time?

In propositional logic we need a copy of each action for each time (situation):

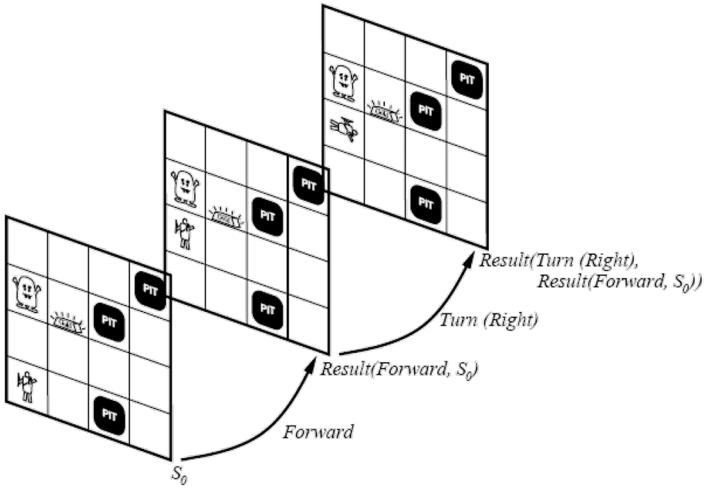
- $L^t_{x,y} \wedge \text{FacingRight}^t \wedge \text{Forward}^t \Rightarrow L^{t+1}_{x+1,y}$
- We need an upper bound for the number of steps to reach a goal but this will lead to a huge number of formulas.

Can we do it better in **first order logic**?

- We do not need copies of axioms describing state changes; this can be implemented using a universal quantifier for time (situation)
- $\forall t$ P is the result of action A in time t+1



- **actions** are represented by terms
  - **Go(x,y)**
  - **Grab(g)**
  - **Release(g)**
- **situation** is also a term
  - initial situation: S_0
  - situation after applying action *a* to state *s*: **Result(a,s)**



- **fluent** is a predicate changing with time
  - the situation is in the last argument of that term
  - **Holding(G, S_0)**
- **rigid** (eternal) **predicates**
  - **Gold(G)**
  - **Adjacent(x,y)**

We need to reason about sequences of actions – about **plans**.

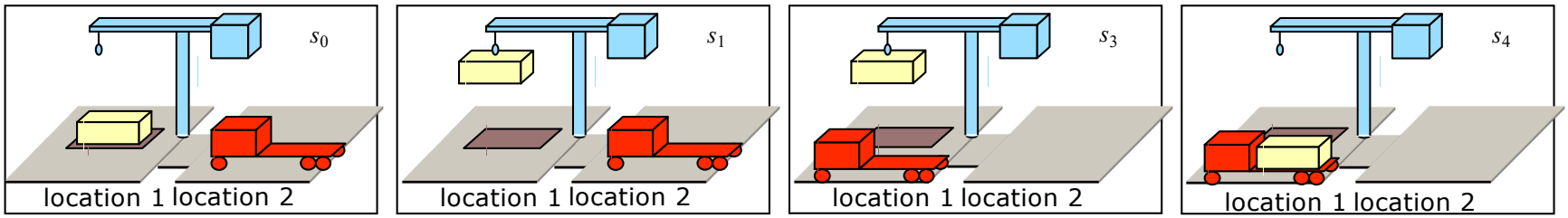
- Result([],s) = s
- Result([a|seq],s) = Result(seq, Result(a,s))

→ [a/seq] je Haskell notace pro první prvek seq. a zbytku

What are typical tasks related to plans?

- **projection task** – what is the state/situation after applying a given sequence of actions?
  - At(Agent, [1,1] , S_0) ∧ At(G, [1,2], S_0) ∧ ¬Holding(o, S_0)
  - At(G, [1,1], Result([Go([1,1],[1,2]),Grab(G),Go([1,2],[1,1])], S_0))
- **planning task** – which sequence of actions reaches a given state/situation?
  - ∃seq At(G, [1,1], Result(seq, S_0))

→ existuje sekvence, která zajistí, že G bude na 1,1, pokud začínáme v S_0?



Situation calculus: actions

Each **action** can be described using two axioms:

- **possibility axiom:** Preconditions $\Leftrightarrow$ Poss(a,s)
  - At(Agent,x,s) $\wedge$ Adjacent(x,y) $\Leftrightarrow$ Poss(Go(x,y),s)
  - Gold(g) $\wedge$ At(Agent,x,s) $\wedge$ At(g,x,s) $\Leftrightarrow$ Poss(Grab(g),s)
  - Holding(g,s) $\Leftrightarrow$ Poss(Release(g),s)
- **effect axiom:** Poss(a,s) $\Rightarrow$ Changes
  - Poss(Go(x,y),s) $\Rightarrow$ At(Agent,y,Result(Go(x,y),s))
  - Poss(Grab(g),s) $\Rightarrow$ Holding(g,Result(Grab(g),s))
  - Poss(Release(g),s) $\Rightarrow$ $\neg$Holding(g,Result(Release(g),s))

protože jsme to definovali pouze pro Agenta

Beware! This is not enough to deduce that a plan reaches a given goal.

We can deduce At(Agent, [1,2], Result(Go([1,1],[1,2]), S_0))
but we **cannot deduce** At(G, [1,2], Result(Go([1,1],[1,2]), S_0))

—> to že agent nějaký došel mi neříká nic o tom, kde je gold

Effect axioms describe what has been changed in the world but they say nothing about the property that everything else is not changed!

This is a so called **frame problem.**

We need to represent properties that are not changed by actions.

A simple **frame axiom** says what is not changed:

$$At(o,x,s) \wedge o \neq Agent \wedge \neg Holding(o,s) \Rightarrow At(o,x,Result(Go(y,z),s))$$

→ pokud objekt není agent a není nikým držen, tak i po GO bude stát na stejném místě

– for F fluents and A actions we need O(FA) frame axioms

– This is a lot especially taking in account that most predicates are not changed.

→ tohle ale vyřeší (zbytečně) hodně axiomů

Frame problem: better axioms

Can we use less axioms to model the frame problem?

- **successor-state axiom**

  Poss(a,s) ⇒
  (fluent holds in Result(a,s) ⇔
  fluent is effect of a ∨ (fluent holds in s ∧ a does not change fluent))

  – We get F axioms (F is the number of fluents) with O(AE) literals in total (A is the number of actions, E is the number of effects).

  Examples:

  Poss(a,s) ⇒
  (At(Agent,y,Result(a,s)) ⇔ a=Go(x,y) ∨ (At(Agent,y,s) ∧ a≠Go(y,z)))

  Poss(a,s) ⇒
  (Holding(g,Result(a,s)) ⇔ a=Grab(g) ∨ (Holding(g,s) ∧ a≠Release(g)))

  Beware of implicit effects!

  - If an agent holds some object and the agent moves then the object also moves.
  - This is called a **ramification problem**.

  Poss(a,s) ⇒
  (At(o,y,Result(a,s)) ⇔
  (a=Go(x,y) ∧ (o=Agent ∨ Holding(o,s))) ∨
  (At(o,y,s) ∧ ¬∃z (y≠z ∧ a=Go(y,z) ∧ (o=Agent ∨ Holding(o,s)))))

Frame problem: even better axioms

Successor-state axiom is still too big with O(AE/F) literals in average.

- To solve the projection task with t actions, the time complexity depends on the total number of actions – O(AEt) – rather than on the actions in plan.
- If we know each action, cannot we do it better, say O(Et)?

classical successor-state axiom:

$$Poss(a,s) \Rightarrow (F_i(Result(a,s)) \Leftrightarrow (a=A_1 \vee a=A_2 \vee \ldots) \vee (F_i(s) \wedge a\neq A_3 \wedge a\neq A_4 \ldots))$$

actions having F_i among effects

actions having $\neg F_i$ among effects

We can introduce **positive** and **negative effects** of actions:

- **PosEffect(a, F_i)** action a causes F_i to become true
- **NegEffect(a, F_i)** action a causes F_i to become false

modified successor-state axiom:

$$Poss(a,s) \Rightarrow (F_i(Result(a,s)) \Leftrightarrow PosEffect(a, F_i) \vee (F_i(s) \wedge \neg NegEffect(a,F_i)))$$

$PosEffect(A_1, F_i)$
$PosEffect(A_2, F_i)$
$NegEffect(A_3, F_i)$
$NegEffect(A_4, F_i)$

© 2020 Roman Barták

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz