Knowledge Representation and Reasoning
Chapter 1. Introduction

Pedro Cabalar

Depto. Computación
University of Corunna, SPAIN

January 30, 2017

Outline

Debate

- Define Knowledge Representation (KR)
- Define Automated Reasoning
- Identify systems or applications where they might be used
- Pros and Cons

The origins of KR

John McCarthy (1927-2011)

- What is Artificial Intelligence?
  "It is the science and engineering of making intelligent machines, especially intelligent computer programs"
- *Programs with Commonsense* [1959] introduces the first AI system: the Advice Taker.
- Keypoint: make an explicit representation of the domain using logical formulae. In McCarthy's words:

  > *"In order for a program to be capable of learning something it must first be capable of being told it"*

Reasoning about Actions and Change (RAC)

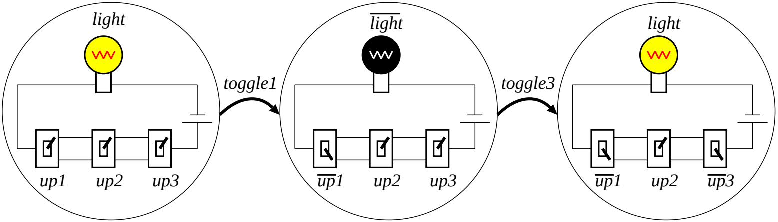
- Thus, under this focusing, Knowledge Representation (KR) plays a central role.
- Historically, one of the first challenging areas for KR has been Reasoning about Actions and Change.
- *Some philosophical problems from the standpoint of Artificial Intelligence* [McCarthy & Hayes 69]

  They introduce Situation Calculus = First Order Logic + 3 *sorts*:

  1. Fluents: system properties whose values may vary along time. These values configure the system state.
  2. Actions: possible operations that allow a state transition.
  3. Situations: terms that identify a given instant. They have the form:
     - ★ S_0 - initial situation
     - ★ $Result(a, s)$ - situation after perfoming action a at situation s.

RAC scenarios

- Typically, (discrete) dynamic systems: state transitions.
- A simple scenario: a lamp in a corridor with 3 switches.
- Fluents: $up1$, $up2$, $up3$, $light$ (Boolean).
- Actions: $toggle1$, $toggle2$, $toggle3$.
- State: a possible configuration of fluent values. Example: $\{\overline{up1}, up2, up3, \overline{light}\}$.
- Situation: a moment in time. We can just use $0, 1, 2, \ldots$

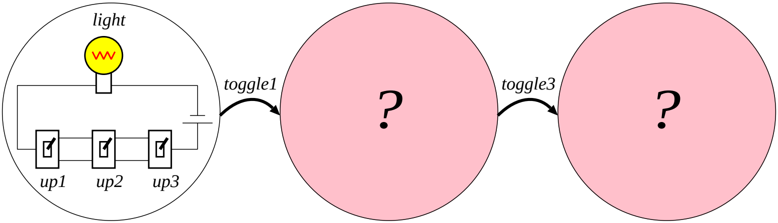


RAC goals

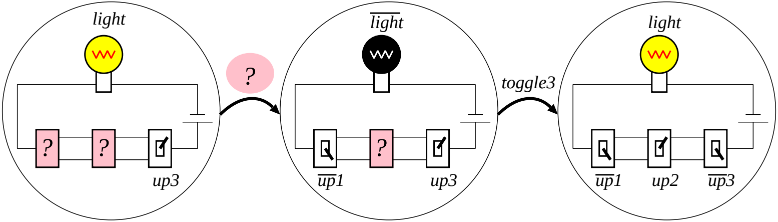
- We want to solve some typical reasoning problems.
- Here is a (non-exhaustive) list of the most usual ones:
  1. Prediction, simulation or temporal projection
  2. Postdiction or temporal explanation
  3. Planning
  4. Diagnosis
  5. Checking properties: system properties; representation properties
- For instance, in our example . . .

Prediction (simulation, or temporal explanation)

- **Knowing**: initial state + sequence of actions
- **Find out**: final state (alternatively sequence of intermediate states)

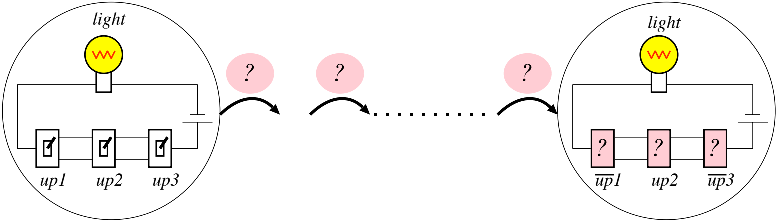


- **Knowing**: partial observations of states and performed actions
- **Find out**: complete information on states and performed actions



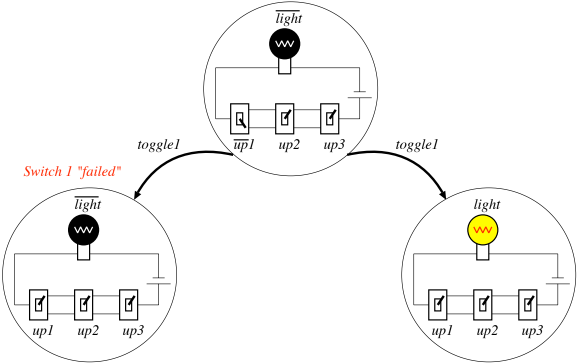
Planning

- **Knowing**: initial state + goal (partial description of final state)
- **Find out**: plan (sequence of actions) that guarantees reaching the goal

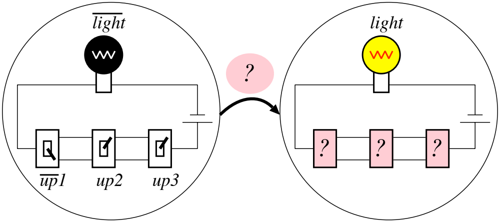


Planning vs Postdiction

- Note that planning seems a type of postdiction. For deterministic systems, this is true, but . . .
- Nondeterministic transition system: fixing current state + performed action $\longrightarrow$ several possible successor states.
- For instance, switch 1 up may fail to turn the light on...



Planning vs Postdiction



- For postdiction, one valid explanation is: we performed *toggle*1, and it succeeded to turn the light on.
- For planning, *toggle*1 is not a valid plan: it does not guarantee reaching the goal *light*. Possible plans are *toggle*2 or *toggle*3.
- Planning in nondeterministic systems is more related to abduction.

Diagnosis

- **Knowing**: a model distinguishing between normal and abnormal transitions + a partial set of observations (usually implying abnormal behavior).
- **Find out**: the minimal set of abnormal transitions that explains the observations.
- Similar to postdiction, but we are additionally interested in minimality of explanations.

Checking properties

- Some elaborated problems are related to checking properties (less common in RAC).
- One may be interested in system properties, as in model checking:
  1. Safety: a given (unsafe) state or condition is never reached. "Something bad never happens"
  2. Liveness: after some condition, something will be eventually reached. "Something good will eventually happen". Example: *any request is eventually attended* Liveness can only be violated with infinite traces.
  3. Fairness: fair resolution of nondeterminism. Example: avoid starvation. *A process turn cannot be infinitely denied*.

Checking properties

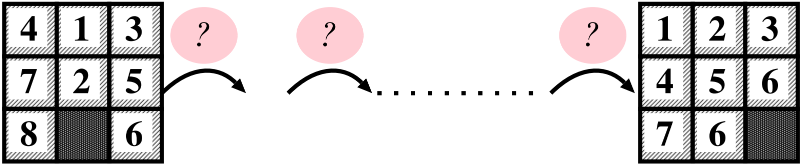
- Or we may be interested in representation properties.
- Are two different representations equivalent? That is, do they generate the same state transition system?
- We will see one stronger concept: are two different representations strongly equivalent? That is, do they generate the same system even when included in a common larger description (or context)?

Example-based methodology

- Paraphrasing McCarthy's comment in a workshop:
  AI researchers start from examples and then try to generalize.
  Philosophers start from the most general case, and never use examples unless they are forced to.
- Advantage: focus on features under study using a synthetic, limited scenario (games, puzzles, etc)
- Real problems usually contain complex factors that happen to be irrelevant for the property under study.

Example-based methodology

- A classical (planning) example: the *N*-puzzle.



- Well known: the 8-puzzle has 181440 sates, the 15-puzzle more than 10^{13}.
- Complexity: NP-complete.

Example-based methodology

- An interesting analogy: “Chess is the Drosophila of AI” [A. Kronrod 65]. That is, games for AI can play the same role as fruit flies for Genetics.
- Warning: avoid too much focus on the toy problem. Remember we must be capable of generalizing the obtained results.
- Back to the chess example:

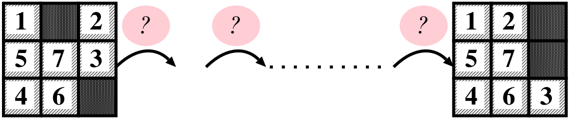
> *“Unfortunately, the competitive and commercial aspects of making computers play chess have taken precedence over using chess as a scientific domain. It is as if the geneticists after 1910 had organized fruit fly races and concentrated their efforts on breeding fruit flies that could win these races.” [McCarthy]*

Example-based methodology

- Take the 8-puzzle example. Which is our main goal? Making a very fast solver for 8-puzzle?
- But what can we learn from that? Which is the application to other scenarios?
- We should perhaps wonder which other scenarios. Originally, AI goal was any scenario (General Problem Solver) but was too ambitious.
- It could perhaps suffice with similar scenarios. Small variations or elaborations.

Elaboration

- Example: assume we may allow now two holes.



- Less steps to solve. We can even allow simultaneous movements.
- Can we easily adapt our solver to this elaboration?
- Think about an optimized heuristic search algorithm programmed in C, for instance.

Keypoint: representation

- A much more flexible solution:
  add a description of the scenario as an input to our solver.
- In this way, variations of the scenario would mean changing the problem description . . . Knowledge Representation (KR) is crucial!

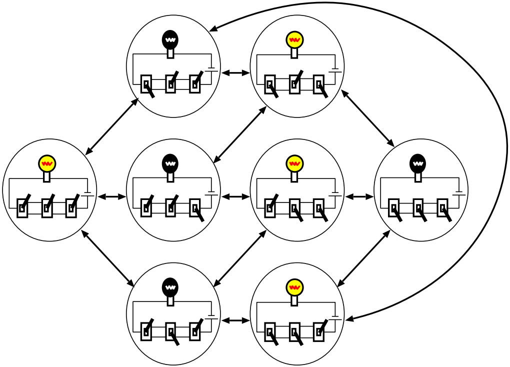
Keypoint: representation

- Which are the desirable properties of a good KR?
  1. Simplicity
  2. Natural understanding: clear semantics
  3. Allows automated reasoning methods that:
     - ★ are efficient
     - ★ or at least, their complexity can be assessed
  4. Elaboration tolerance [McCarthy98]

     "A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances." [McCarthy98]

Keypoint: representation

- Example of representation: an automaton is simple, and has a clear semantics . . .
- But fails in elaboration tolerance! A small change (say, adding new switches or lamps) means a complete rebuilding



Keypoint: representation

- A practical alternative: use rules to describe the local effects of each performed action.
- For each switch $X \in \{1, 2, 3\}$

$$\begin{array}{rcl}
\text{Action} & \text{precondition} & \Rightarrow \text{effect(s)} \\
\hline
\\
\mathit{toggle}(X): & \mathit{up}(X) & \Rightarrow \overline{\mathit{up}(X)} \\
\mathit{toggle}(X): & \overline{\mathit{up}(X)} & \Rightarrow \mathit{up}(X) \\
\mathit{toggle}(X): & \mathit{light} & \Rightarrow \overline{\mathit{light}} \\
\mathit{toggle}(X): & \overline{\mathit{light}} & \Rightarrow \mathit{light}
\end{array}$$

- This language is similar to STRIPS [Fikes & Nilsson 71] still used in planning systems.

- Can we just use classical logic instead?

$$toggle(X) : up(X) \quad \Rightarrow \quad \overline{up(X)}$$
$$toggle(X, I) \wedge up(X, true, I) \quad \rightarrow \quad up(X, false, I+1)$$

where we include as new arguments, the temporal indices $I, I+1$ plus the fluent values *true*, *false*.

- Problem: when $toggle(1)$, what can we conclude about $up(2)$ and $up(3)$?
They should remain unchanged! However, our logical theory provides no information (we also have models where their value change).

- We would need much more formulae

$$\begin{array}{rcl} toggle(1, I) \wedge up(2, true, I) & \rightarrow & up(2, true, I+1) \\ toggle(1, I) \wedge up(2, false, I) & \rightarrow & up(2, false, I+1) \end{array}$$

$$\begin{array}{rcl} toggle(1, I) \wedge up(3, true, I) & \rightarrow & up(3, true, I+1) \\ toggle(1, I) \wedge up(3, false, I) & \rightarrow & up(3, false, I+1) \\ & \vdots & \end{array}$$

and so on, for any fluent and value that are unrelated to $toggle(1)$.

Default reasoning

- Frame problem: adding a simple fluent or action means reformulating all these formulae! [McCarthy & Hayes 69]
- We need a kind of default reasoning.
  Inertia rule: fluents remain unchanged *by default*
- "By default" = when no evidence on the contrary is available. We must extract conclusions from absence of information.
- Unfortunately, Classical Logic is not well suited for this purpose because

  $$\Gamma \vdash \alpha \text{ implies } \Gamma \cup \Delta \vdash \alpha$$

  This is called monotonic consequence relation.
- But $\Gamma \vdash \alpha$ by default could mean that adding Δ, $\Gamma \cup \Delta \not\vdash \alpha$.
  We need Nonmonotonic Reasoning (NMR).

Default reasoning

- An example: suppose $up(2, true, 0)$ and we perform $toggle(1, 0)$. Inertia should allow us to conclude that switch 2 is unaffected:

$$\Gamma \vdash up(2, true, 1)$$

- Elaboration: we are said now that $toggle(1)$ affects $up(2)$ in the following way:

$$toggle(1, I) \wedge up(2, true, I) \rightarrow up(2, false, I + 1) \tag{1}$$

  We will need retract our previous conclusion

$$\Gamma \cup (1) \not\vdash up(2, true, 1)$$

Other typical representational problems

- Qualification problem: preconditions are affected by conditions that qualify an action.
- Example: when can we toggle the switch? Elaborations: switch is not broken, switch has not been stuck, we must be close enough, etc.
- The explicit addition of any imaginable “disqualification” is unfeasible. Again: by default, toggle works when nothing prevents it.

Other typical representational problems

- Elaboration: there is a light sensor that activates an alarm, if the latter is connected. The alarm causes locking the door.
- In STRIPS, this means relating indirect effects *alarm* to each possible action $toggle(X)$.

$$\begin{array}{lll} \text{Action} & \text{precondition} & \Rightarrow \text{effect(s)} \\ \hline toggle(X): & \overline{light}, connected & \Rightarrow alarm \\ toggle(X): & \overline{light}, connected & \Rightarrow lock \end{array}$$

Problem: there may be other new ways to turn on a light, or to activate the alarm. We will be forced to relate *lock* to the performed actions!

Other typical representational problems

- This is called ramification problem: postconditions are affected by interactions due to indirect effects.
- *lock* is an indirect effect of toggling a switch ($toggle \mapsto light \mapsto alarm \mapsto lock$).
- We would need something like:

$$\begin{aligned} light(true, I) \wedge connected(true, I) &\rightarrow alarm(true, I) \\ alarm(true, I) &\rightarrow lock(true, I) \end{aligned}$$

Outline

KR is a well-established field

Main conferences including KR

- KR: Intl. Conf. on Principles of Knowledge Representation and Reasoning $\Rightarrow$ last edition `http://www.kr.org/KR2016/`
- IJCAI: Intl. Joint Conf. on Artificial Intelligence
- AAAI: Conf. on Artificial Intelligence
- JELIA: European Conf. on Logics in Artificial Intelligence
- LPNMR: Intl. Conf. on Logic Programming and Non-Monotonic Reasoning   LPNMR'13 cellebrated in Corunna!
- Workshop on Logical Formalizations of Commonsense Reasoning

KR is a well-established field

These are some of the usual topics in KR call for papers:

- Argumentation
- Belief revision and update
- Computational aspects of KR
- Contextual reasoning
- Description logics
- Explanation finding, diagnosis, causal reasoning, abduction
- Inconsistency and exception tolerant reasoning, paraconsistent logics
- Logical approaches to planning and behavior synthesis
- Logic programming, answer set programming, constraint logic programming
- Nonmonotonic logics, default logics, conditional logics
- Reasoning about norms and organizations, social knowledge and behavior ...

KR is a well-established field

…

- Philosophical foundations of KR
- Ontology languages and modeling
- Preference modeling and representation, reasoning about preferences, preference-based reasoning
- Qualitative reasoning, reasoning about physical systems
- Reasoning about actions and change, action languages, situation calculus, dynamic logic
- Reasoning about knowledge and belief, epistemic and doxastic logics
- Spatial reasoning and temporal reasoning
- Uncertainty, representations of vagueness, many-valued and fuzzy logics, relational probability models

KR is a well-established field

Interesting link for further reading:

http://aitopics.org/topic/representation-reasoning