Knowledge Based Systems

Jacek Malec
Dept. of Computer Science, Lund University, Sweden
February 25, 2014

Programming vs KR

Programming:

- construct an algorithm to solve a problem
- choose a programming language
- implement the algorithm in the language
- run the program to solve the problem

Knowledge Representation and Engineering:

- identify knowledge necessary to solve the problem
- choose a representation language
- formulate knowledge in the language
- check if solution can be derived

Knowledge

There is evidence that people memorize just **the important stuff**.

- objects
- facts, relations
- rules (generic knowledge)
- causality
- meta-knowledge
- procedural knowledge
- knowledge about processes
- ...

Meta-knowledge

knowledge about other's (or our own)

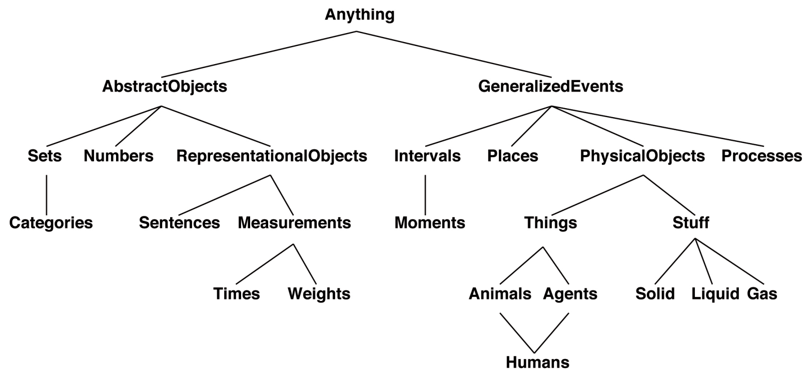
- knowledge (Jacek knows that Earth revolves around Sun)
- beliefs (Jacek believes that his daughter is smart)
- needs (Jacek needs more time for checking assignments)
- goals (Jacek wants to write a paper by Friday)
- intentions (Jacek wants to be ready with assignment 1 by tomorrow)
- capabilities (Jacek can drive)
- ...

Procedural knowledge

How to:

- change a diaper?
- serve in tennis?
- ride a bicycle?
- drive a car?
- ...

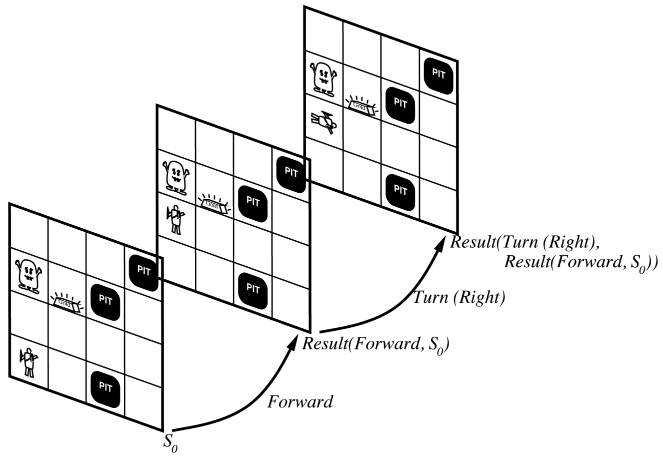
Ontology



Problems with predicate logic:

- static
- flat
- qualification/ramification/frame problem
- exceptions
- strength
- ...

Actions, situations



Situation calculus

Action descriptions:

- possibility axioms (when is an action possible)
- effect axioms (what's its effect, what changes)
- frame axioms (what remains the same)
  Important issue!

Quite often we need richer ontology.

Describing actions

"Effect" axiom—describe changes due to action

$$\forall s\ AtGold(s) \Rightarrow Holding(Gold, Result(Grab, s))$$

"Frame" axiom—describe *non-changes* due to action

$$\forall s\ HaveArrow(s) \Rightarrow HaveArrow(Result(Grab, s))$$

"Successor-state axioms" solve the representational frame problem

Successor-state axioms

Each axiom is "about" a *predicate* (not an action per se):

P true afterwards	$\Leftrightarrow$	[an action made P true
	$\vee$	P true already and no action made P false]

For holding the gold:

$$\forall a, s\ Holding(Gold, Result(a, s)) \Leftrightarrow$$

$$[(a = Grab \wedge AtGold(s)) \vee (Holding(Gold, s) \wedge a \neq Release)]$$

Making plans

Initial condition in KB:
$At(Agent, [1, 1], S_0)$
$At(Gold, [1, 2], S_0)$

Query: $Ask(KB, \exists s\ Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s | Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Making plans: A better way

Represent *plans* as action sequences $[a_1, a_2, \ldots, a_n]$

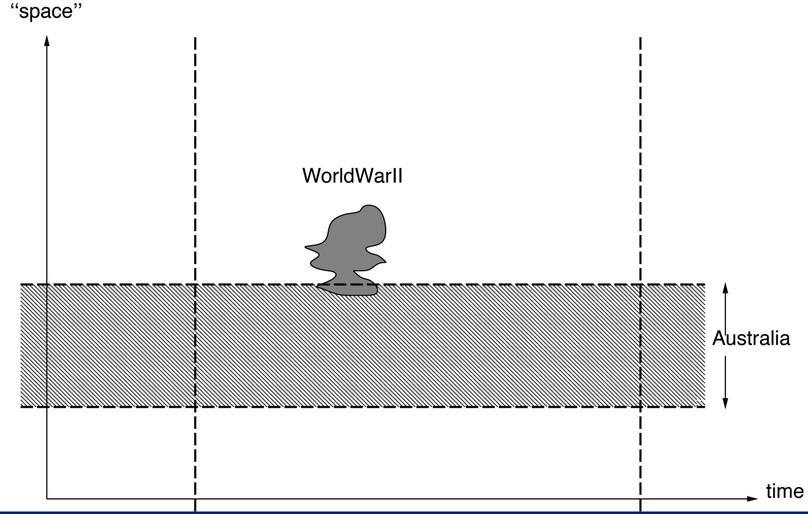
$PlanResult(p, s)$ is the result of executing p in s

Then the query $Ask(KB, \exists p\ Holding(Gold, PlanResult(p, S_0)))$ has the solution $\{p|[Forward, Grab]\}$

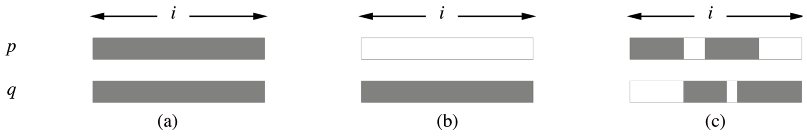
Definition of *PlanResult* in terms of *Result*:
 $\forall s\ PlanResult([\,], s) = s$
 $\forall a, p, s\ PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

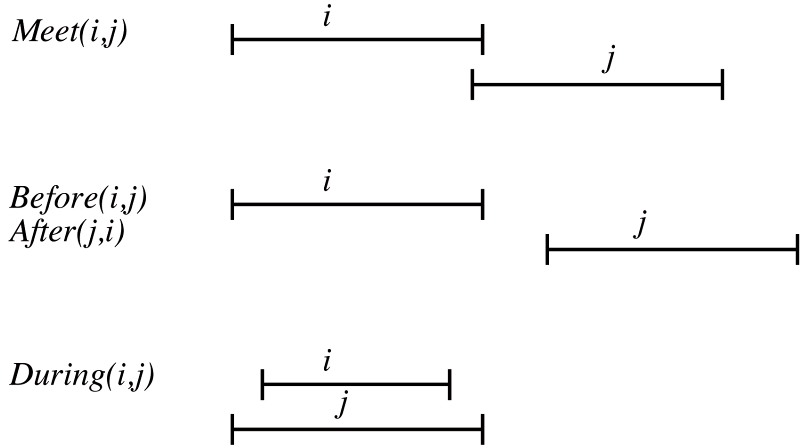
Events



Events



Interval calculus



Representation of exceptions:

$\forall x(Bird(x) \wedge \neg Pinguin(x) \wedge \neg Ostrich(x) \rightarrow Flies(x))$
But if we know $Bird(Tweety)$ we can't say whether it flies!

Qualification Problem:

Unfortunately, many other birds don't fly either: caged birds, dead birds, birds with broken wing, bird with feathers covered by oil, ...
$\forall x(Bird(x) \wedge \neg Pinguin(x) \wedge \neg Ostrich(x) \wedge \ldots \rightarrow Flies(x))$

Other related problems: frame problem, ramification problem

Solutions:

- non-monotonic logics
- fuzzy logic
- probabilistic logic
- possibilistic logic
- ...

Monotonic vs. non-monotonic

$Th(\Delta)$

monotonic logic:
if $\Delta \subseteq \Delta'$ then $Th(\Delta) \subseteq Th(\Delta')$

non-monotonic logic:
if $\Delta \subseteq \Delta'$ and $Th(\Delta) \supset Th(\Delta')$

Closed World Assumption:

Things that cannot be proven true are probably false

leads to:

Negation as failure

Consider:

$$\Delta = \{A(Stockholm), A(Sturup), A(Copenhagen), A(Oslo), Conn(Oslo, Copenhagen), Conn(Stockholm, Oslo), \forall(x,y,z)(Conn(x,y) \wedge Conn(y,z) \rightarrow Conn(x,z))\}$$

Because it's not the case that $Conn(Sturup, Stockholm)$, CWA gives us immediately $\neg Conn(Sturup, Stockholm)$.
But let us add $Conn(Sturup, Copenhagen)$ to Δ.
$Conn(Sturup, Stockholm)$ can be proven now, so
$\neg Conn(Sturup, Stockholm)$ does not appear as a consequence.
Non-monotonicity!

CWA is syntax-dependent:

If $\Delta = \{Single(John), Single(Mary), Clever(Kent)\}$
then CWA gives us:
$\{\neg Clever(John), \neg Clever(Mary), \neg Single(Kent)\}$.

But if $\Delta = \{\neg Married(John), \neg Married(Mary), Clever(Kent)\}$
then CWA gives us:
$\{\neg Clever(John), \neg Clever(Mary), \neg Married(Kent)\}$.

Effective representation of knowledge:

- storing
- retrieval
- modification

What should be represented?

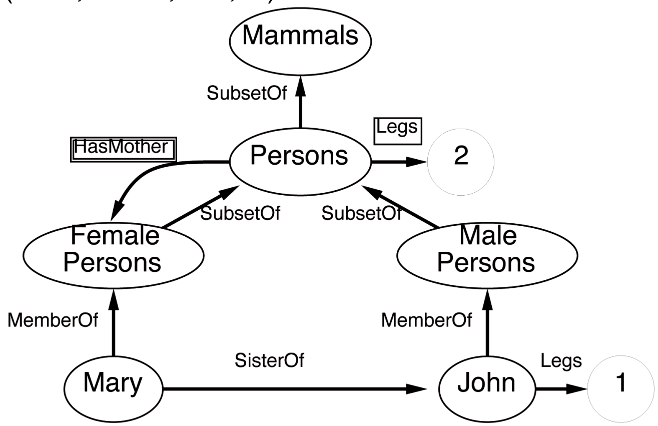
- use logic
- not necessarily FOL

How should it be represented?

- whatever method you find suitable

Semantic networks:

Precursor of Description Logics and Semantic Web Languages (OWL, DAML, OIL, ...)



Reasoning in SN:

- inheritance via SubsetOf (SubClass) and MemberOf (isA) links
- intersection paths
- special meaning associated with some links (like cardinality constraints, etc.)
- classification, consistency, subsumption

May be effective given some restrictions on the logic (DL variants).
Rule-based reasoning on top (RIF and co.)

Knowledge-Based Systems

A generic term, might denote anything that involves encoded knowledge.
Or might mean a system where the knowledge component is *explicit* and manipulable.

Paradigms throughout history of AI:

- Logic-based systems;
- Rule-based systems (expert systems);
- Blackboard systems;
- Semantic web systems.

Rule-based systems

Or *expert systems*.
Promised much. Delivered (too) little. Caused “AI Winter” in the 90s.
Simple architecture:

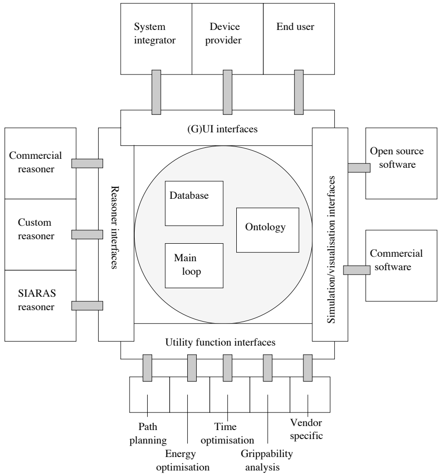
- Facts;
- Rules;
- Inference engine:
  - Matching;
  - Conflict resolution;
  - Rule application.

Blackboard systems

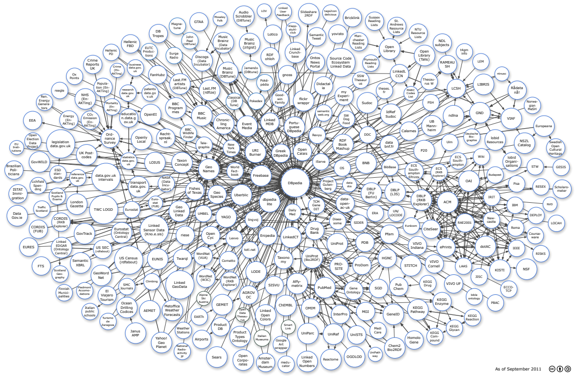
Architecture:

- Knowledge base (blackboard);
- Knowledge sources (expert problem solvers);
- Controller (agenda maintainer).

A blackboard system



Semantic Web



Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. `http://lod-cloud.net/`

Semantic Web

Lots of hype. Lots of acronyms. But some are important!

- URI – Uniform Resource Identifier
- RDF – Resource Description Framework
- RIF – Rule Interchange Format
- SPARQL – SPARQL Protocol and RDF Query Language
- OWL – Web Ontology Language

Open World Assumption!

SPARQL

W3C Recommendation
Queries:

- SELECT (returns a table)
- CONSTRUCT (returns RDF)
- ASK (returns a boolean)
- DESCRIBE (freedom)

Example:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

A SPARQL query

What are all the country capitals in Africa?

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
     abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
     abc:isInContinent abc:Africa .
}
```

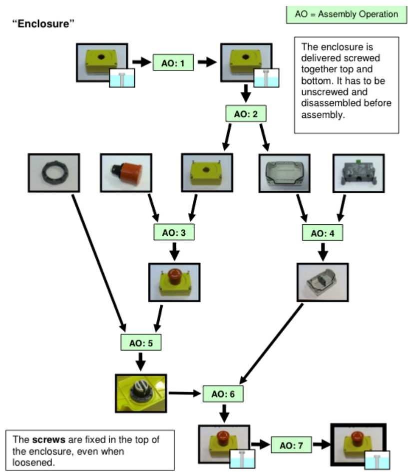

A real SPARQL query

http://wiki.dbpedia.org/OnlineAccess
http://asimov.ludat.lth.se

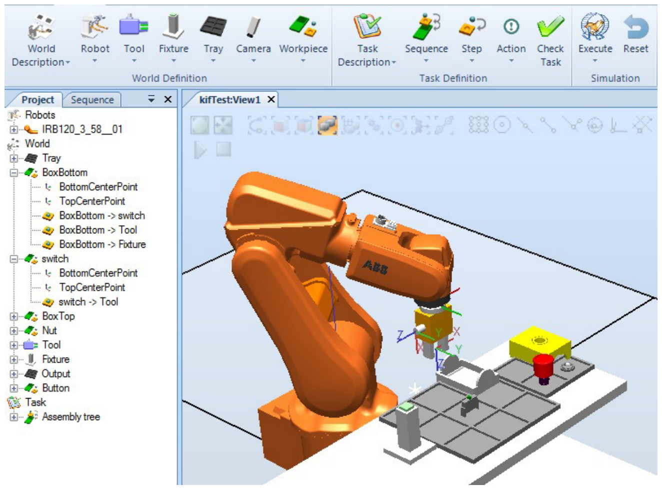
```
select ?s where {
  ?s a rosetta:Camera.
}

select distinct ?s ?v where {
?s a rosetta:Camera.
?s ?p ?n.
?n caex-xml:hasName "FocusRange".
?n caex-xml:hasValue ?v.
}
```

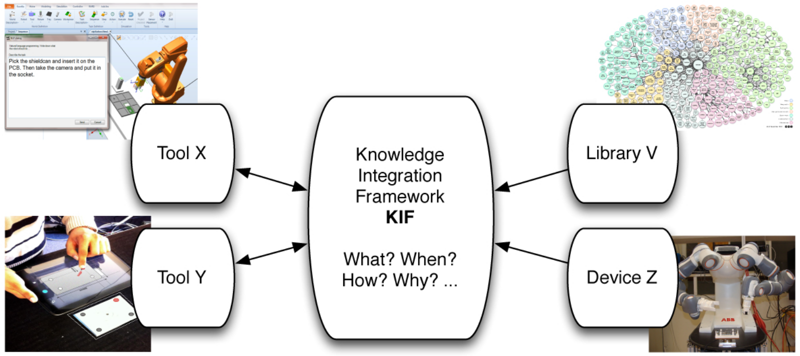

A real case



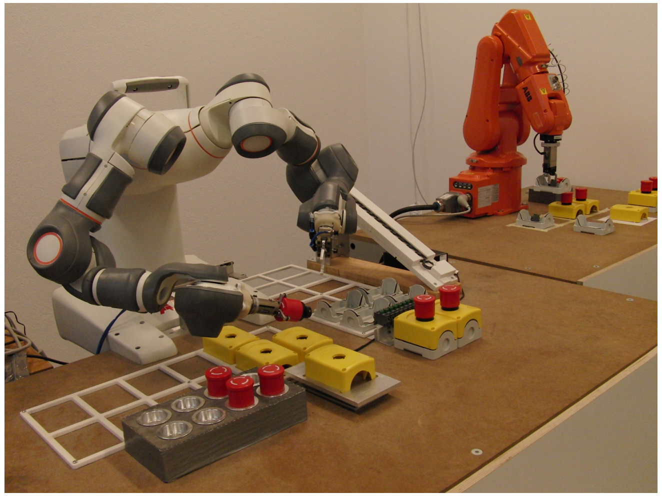
A real case



A real case



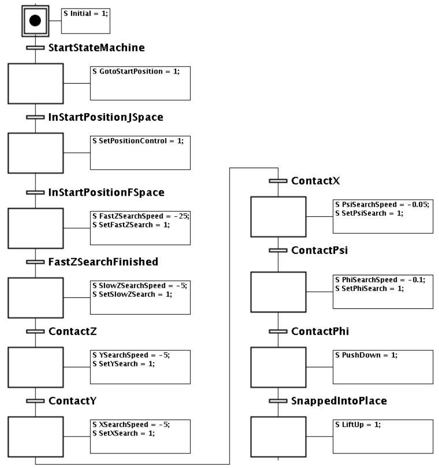
A real case



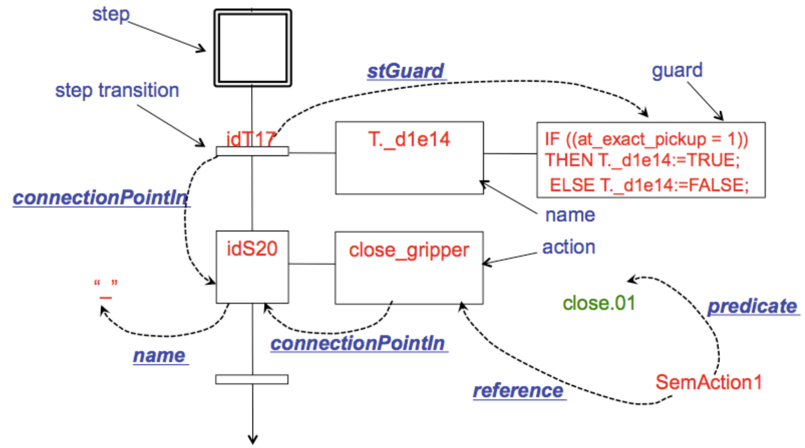
A real case

A real case

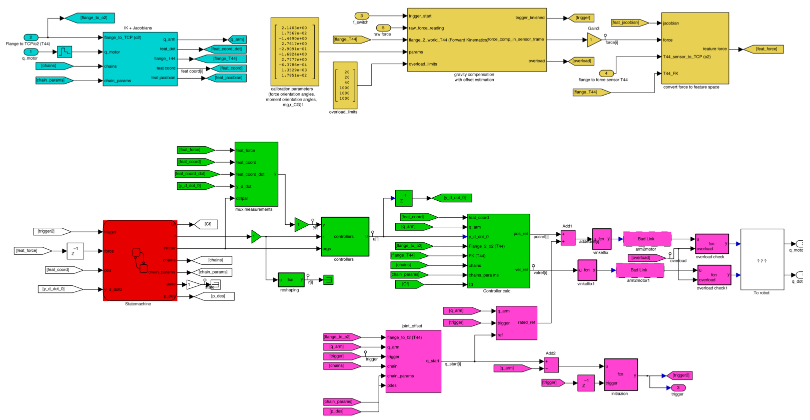
A real case



A real case



A real case



Next period: EDAN50

EDAN50 Intelligent Systems: Project

- topics proposed by us (PN, EAT, JM)
- but: you may choose your own
- personal supervision
- research-related problems
- workload: normal course, 7.5 hp
- Last chance this year!