Introduction to AI

Some applications of KRR

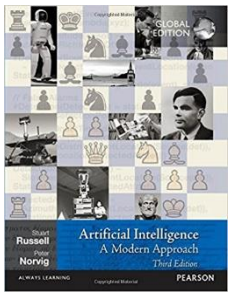
Francesca Toni

(thanks to Fariba Sadri)

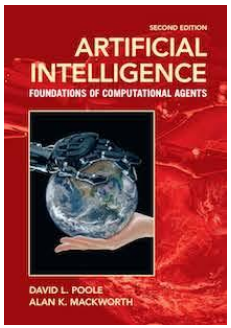
Outline

- Logical agents
- Logic-based Production System
- (Rule-based robotics)
- Ontologies

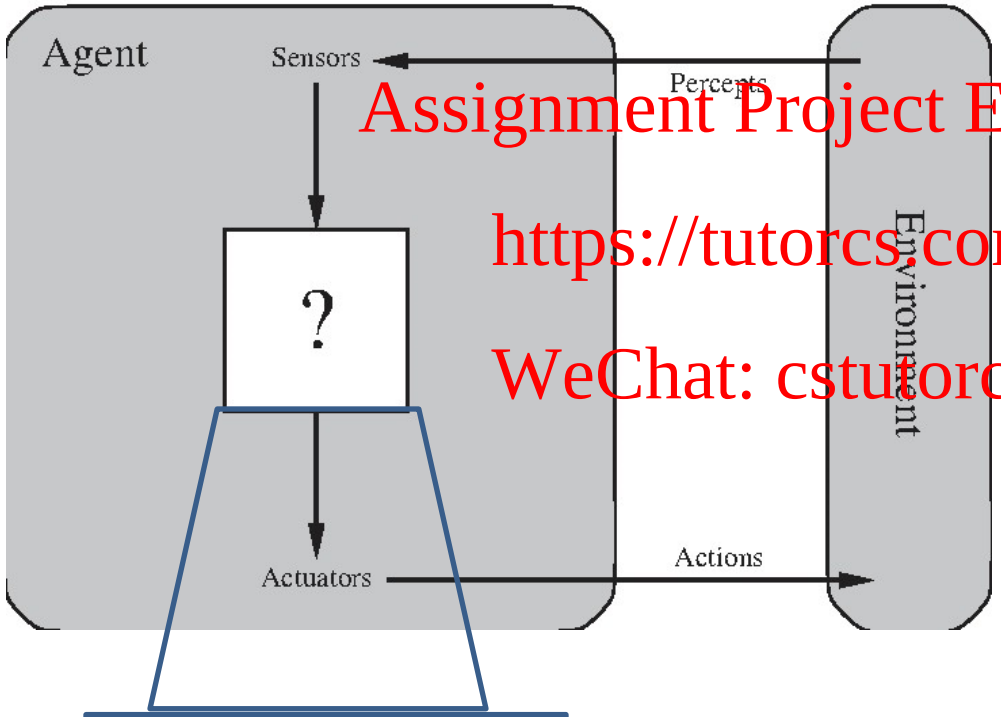
Section 7.1
Section 12.5

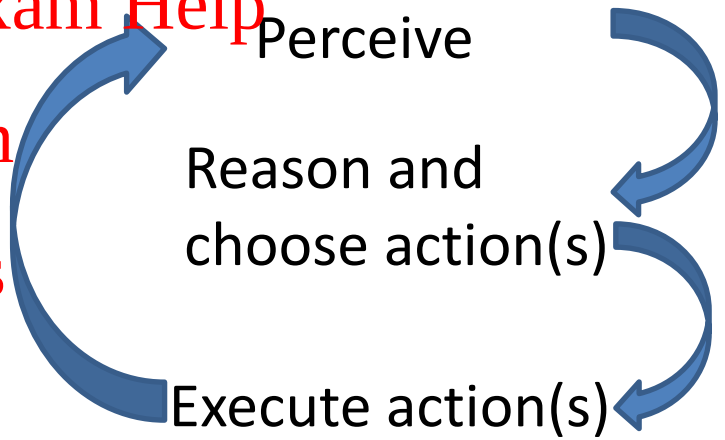


Section 2.4
Section 14.3



Logical agents





- inference engine — Domain-independent/static – e.g. SLDNF/(XSB)Prolog
- knowledge base — Domain-dependent, dynamic – e.g. logic program
- goal(s) — Domain-dependent, dynamic

LPS (Logic-based Production System)

http://lpsdemo.interprolog.com/

Declarations

```
maxTime( ).
fluents    … .
actions    … .
events     … .
```

Inputs

```
initially  … .
observe    … .
```


Reactive rules

```
if … then … .
```

Clauses

```
… if … .
… :- … .
```

Causal theory

```
… initiates … if … .
… terminates … if … .
false … .
```

```
% Fire example
% Declarations
maxTime(5).
fluents     fire.
actions     eliminate, escape.
events      deal_with_fire.

% Initial state: Inputs
initially    fire.

% Goals: Reactive Rules
if fire at T1  then deal-with-fire from T1 to T2.

% Beliefs: Clauses
deal-with-fire from T1 to T2 if eliminate from T1 to T2.
deal-with-fire from T1 to T2 if escape from T1 to T2.

% Causal theory
eliminate  terminates fire.
```

```
% Fire example extended with  recurrent fires
% Declarations
maxTime(10).
fluents     fire, water.
actions     eliminate,  escape, ignite(_), refill.
events      deal_with_fire.

% Initial state: Inputs
initially   water.
% Observations: Inputs
observe   ignite(sofa) from 1 to 2.
observe   ignite(bed) from 4 to 5
observe   refill from 7 to 8.

% Goals: Reactive Rules
if fire at T1  then deal-with-fire from T2 to T3.

% Beliefs: Clauses
deal-with-fire from T1 to T2
   if eliminate from T1 to T2.
deal-with-fire from T1 to T2
   if escape from T1 to T2.
% Time-independent information: Clauses
flammable(sofa).
flammable(bed).
% Causal theory
ignite(Object)  initiates  fire
 if  flammable(Object).
eliminate terminates fire.
eliminate terminates water.
refill initiates water.

false eliminate, fire, not water.
```

```
% Planning in the blocks' world

maxTime(10).
fluents location(_,_).
actions move(_,_).

initially location(f,floor), location(b,f),location(e,b),
location(a,floor), location(d,a),location(c,d).

% Goals
if true
then make_tower([a,b,c,floor]) from T1 to T2.

if true
then make_tower([f,e,d,floor]) from T1 to T2.

clear(Block) at T if Block \= floor,
   not location(_,Block) at T.

clear(floor) at _.
```

```
make_tower([Block,floor]) from T1 to T2 if
   make_on(Block,floor) from T1 to T2.

make_tower([Block,Place|Places]) from T1 to T3 if
            Place \= floor,
   make_tower([Place|Places]) from T1 to T2,
   make_on(Block,Place) from T2 to T3.

make_on(Block,Place) from T1 to T4 if
   not location(Block,Place) at T1,
   make_clear(Place) from T1 to T2,
   make_clear(Block) from T2 to T3,
   move(Block,Place) from T3 to T4.

make_on(Block,Place) from T to T if
   location(Block,Place) at T.

make_clear(Place) from T to T if   clear(Place) at T.

make_clear(Block) from T1 to T2if
   location(Block1,Block) at T1,
   make_on(Block1,floor) from T1 to T2.


move(Block,Place)  initiates location(Block,Place).
move(Block,_)  terminates location(Block,Place).
```

Rule-based systems for robotics (1) -non-examinable-

Baxter in blocks-world

A Framework for Integrating Symbolic and Sub-symbolic Representations

a robot building towers of blocks, subject to human interference, using
1) a concurrent multi-tasking teleo-reactive program,
2) a physics simulator to provide spatial knowledge,
3) sensor processing and robot control.

Rule-based systems for robotics (2) -non-examinable-

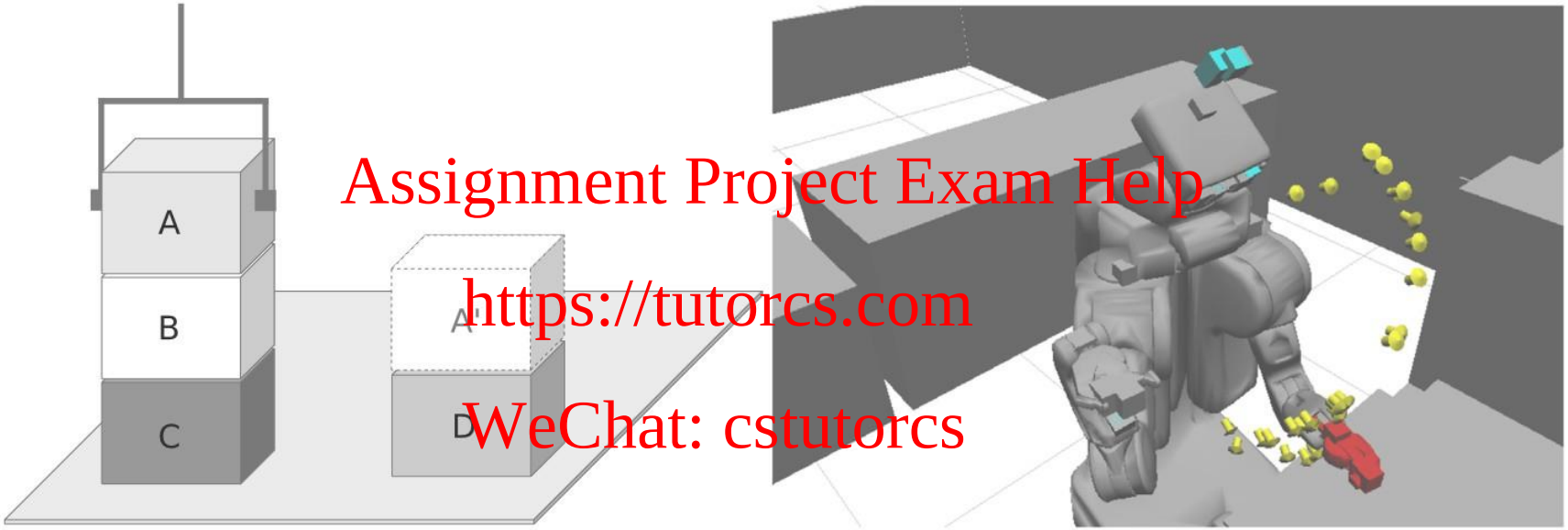
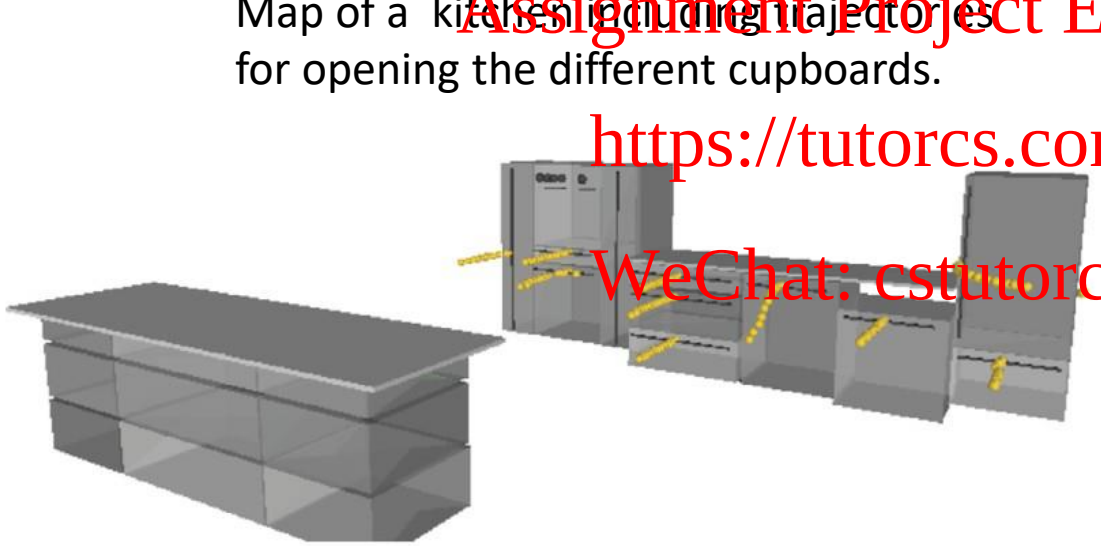


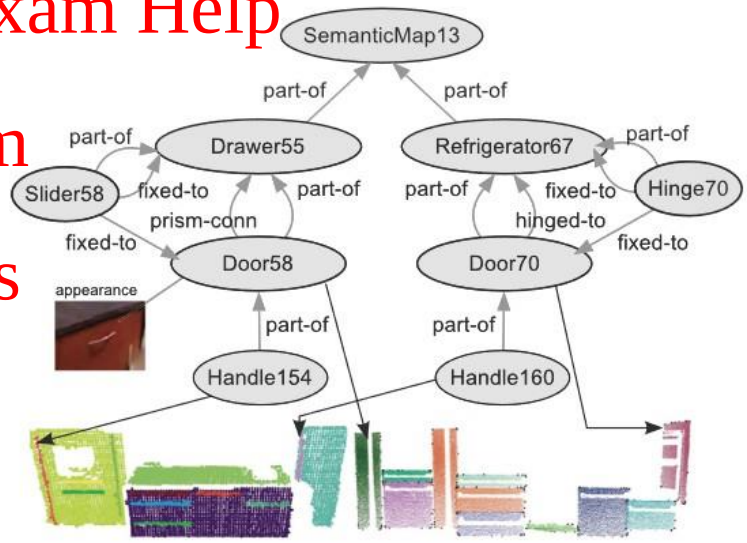
Fig. 1. Left: Example situation in a classical 'blocks world' environment. Using logical inference on the scene representation, the system can determine whether all preconditions are fulfilled for moving block *A* to position *A'*. Right: Visualization of a scene model from the KNOWROB robot knowledge base. While the situation is much more complex, efficient algorithms exist for computing e.g. stability or reachability.

Representations for robot knowledge in the KnowRob framework

Rule-based systems for robotics (2) Semantic environment maps

Map of a kitchen including trajectories for opening the different cupboards.





Ontology: representation of composed objects and kinematic structures with prismatic and rotational joints

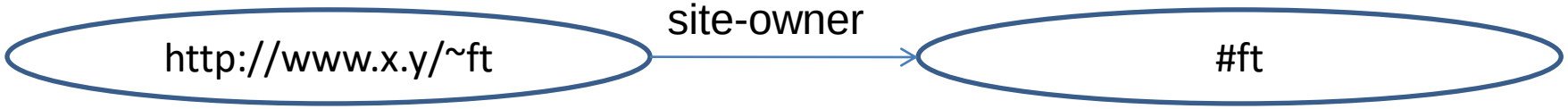
Ontologies

- Explicit and formal specification of a conceptualization …"the kind of things that exist in a given domain"
- Typical Components of Ontologies
  - Concepts of the domain
    - e.g. classes of objects
      - professors, students, courses
  - Relationships between these terms:
    - e.g. class hierarchies: a class C to be a subclass of another class C' if every object in C is also included in C'
      - all professors are staff members
- On the web: ontologies provide a shared understanding of a domain: **semantic interoperability**
  - overcome differences in terminology
  - map between ontologies

RDF (Resource Description Framework):

Universal language for describing resources:

- Statements of the form object-attribute-value assert properties of objects (resources):
  - they consist of an object (resource), a property, and a value
    - e.g. *(http://www.x.y/~ft, http://www.k.z/site-owner, #ft)*
- Can be also seen as
  - a (piece of a) graph (known in AI as a *semantic net*)



  - a piece of XML code
  - an atom: *site-owner(http://www.x.y/~ft, #ft)*

RDF schema

- Classes (and properties)
  - **type(a,C)** states that a is instance of class C

- Class Hierarchies (and Inheritance)
  - **subClassOf(C,D)** states that class C is a subclass of class D

- Property Hierarchies
  - **subPropertyOf(P,Q)** states that property Q is true whenever property P is true

RDF and RDF Schema in positive logic programming (description logic programming)

- A triple of the form **(a,P,b)** in RDF can be expressed as a fact **P(a,b)**
  - E.g. isTaughtBy(introAI,ft).
- An instance declaration of the form **type(a,C)** (stating **a** is instance of class **C**) can be expressed as **C(a)**
  - E.g. professor(ft)
- The fact that **A** is a subclass (or subproperty) of **B** can be expressed as **A(X) → B(X)**
  - E.g. academicStaffMember(X)<-professor(X)

    (involves(X,Y) <- isTaughtBy(X,Y))

Summary

- Knowledge representation (and automated reasoning) for
  - Logical agents (and rule-based robots)
  - Ontologies in RDF and RDF schema - for robots and the web