# TDDE05 AI Robotics
# Knowledgeable Robot

*Cyrille Berger*

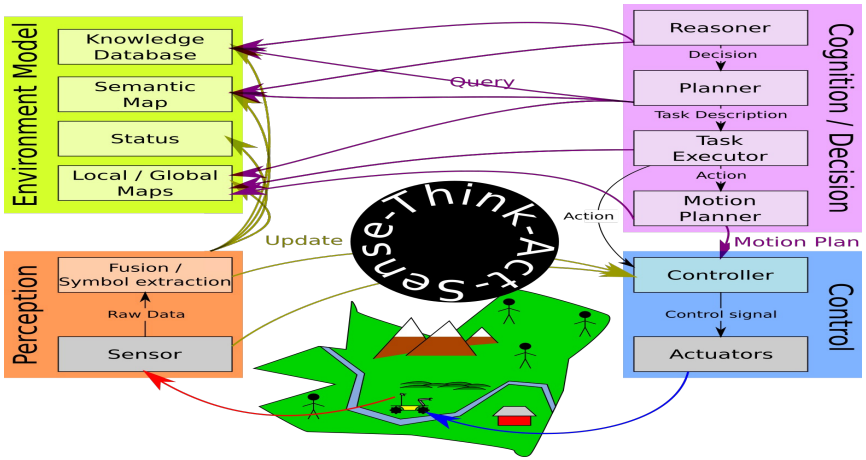
LINKÖPING UNIVERSITY

# Lectures

1 Introduction to AI Robotics
2 Introduction to ROS
3 Foundation of Robotics: Control and State Machines
4 Foundation of Robotics: Perception
5 Robotic Architectures
6 Reactive Architecture
7 Deliberative Architecture
8 Decision Theory
9 **Knowledgeable Robot**
10 Machine Learning for Robots
11 Human-Aware Robots
12 Putting It All Together



# Robot Architecture






# Lecture content

- Knowledge Database
  - Description Logics
  - RDF and the Semantic Web
  - SPARQL
  - Ontologies
    - OWL
  - Ontologies for Robotics

# Knowledge Database



# How to store symbols?

- Symbols have properties
  - The car's color is red
  - The victim's leg is broken
- Symbols have relations to each other
  - The glass is on this table
  - The victim is in that building
- Symbols have generic properties
  - This object is a car, a car has four wheels and can be used to drive humans around

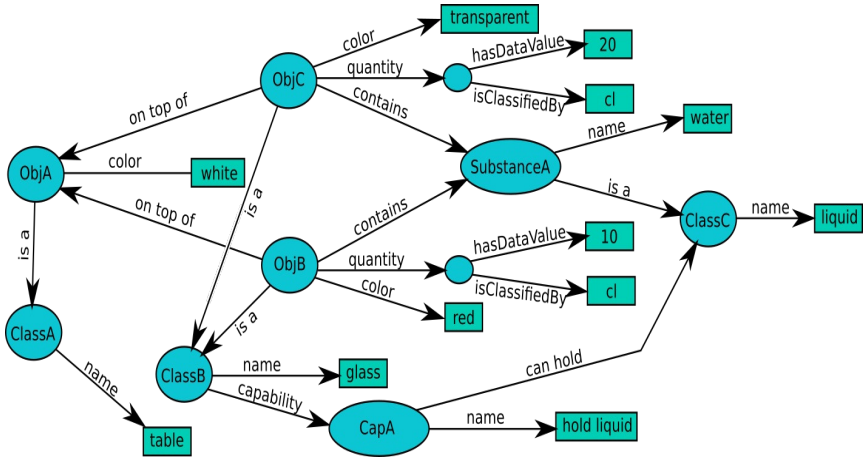


# How to access symbols?

- Where are victims with broken legs?
- Where is the glass?
- What can be used to drive humans around?



# Modeled as a graph

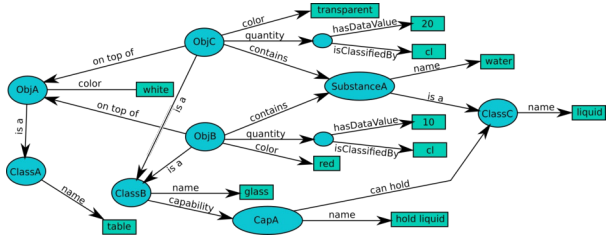


# Description Logics



# What are Description Logics (DL)?

- A family of logic based on Knowledge Representation formalisms
- Describes domain in terms of *concepts* (aka classes), *roles* (aka properties, relationships) and *individuals*




- A subset of first-order logic, that is decidable



# Knowledge base

Knowledge is represented as a knowledge base, $K = \langle A, T(, R) \rangle$ where:

- T (TBox) is the **Terminological Knowledge**
  - Knowledge about concepts of a domain
  - Victim ≡ Person ⊓ ∃hasStatus.Injured
- A (ABox) is the **Assertion Knowledge**
  - Knowledge about individuals / Entities
  - Person(George)
  - hasStatus(Injured, George)
- R (RBox) is the **Role-centric Knowledge**
  - Person ⊑ Mammals



# ALC

- ALC (Attribute Language with Complement) is the smallest deductively complete DL
  - Class constructors: conjuction ⊓, disjunction ⊔, negation ¬
  - Quantifiers restricted to domain and range of roles

# ALC: Example

- Individuals
  - Individual *ObjA* is of class *Table*
  - Table(ObjA)
  - Individual *ObjB* is of class *Glass*
  - Glass(ObjA)
- Roles
  - *ObjB* is *on top of ObjA*
  - onTopOf(ObjB, ObjA)



# ALC: Formal Definition

- Set of concept A, B, C...
- Special concepts: universal (⊤), empty (⊥)
- Set of role names R, S...
- Set of individuals a,b,c...
- Complex classes are defined with
- C,D:=A | ⊤ | ⊥ | ¬C | C⊓D | C⊔D | ∀R.C | ∃R.C
- A TBox is a set of statements of the form C≡D or C⊑D
- A ABox is a set of statements of the form C(a) or R(a,b)



# Examples

- TBox
  - Animal ⊑ LivingThing
  - Donkey ≡ Animal ⊓ ∀hasParent.Donkey
  - Horse ≡ Animal ⊓ ∀hasParent.Horse
  - Mule ≡ Animal ⊓ ∃hasParent.Horse ⊓ ∃hasParent.Donkey
  - ∃hasParent.Mule ⊑ ⊥
- ABox
  - Horse(Mary) Mule(Peter ) Donkey (Sven)
  - hasParent(Peter , Mary ) hasParent(Peter , Carl)



# Translation to First order logic (1/2)

- conjunction

$$Mother \equiv Woman \sqcap Parent$$
$$\forall x \quad Mother(x) \Longleftrightarrow Woman(x) \wedge Parent(x)$$

- disjunction

$$Parent \equiv Mother \sqcup Father$$
$$\forall x \quad Parent(x) \Longleftrightarrow Mother(x) \vee Father(x)$$

- negation

$$ChildlessPerson \equiv Person \sqcap \neg Parent$$
$$\forall x \quad ChildlessPerson(x) \Longleftrightarrow Person(x) \wedge \neg Parent(x)$$

# Translation to First order logic (2/2)

- existential qualification

$$Parent \equiv \exists hasChild.Person$$
$$\forall x \quad Parent(x) \iff \exists y \quad hasChild(x, y) \wedge Person(y)$$

- universal qualification

$$Person \sqcap Happy \equiv \forall hasChild.Happy$$
$$\forall x \quad Person(x) \wedge Happy(x) \iff (\forall y hasChild(x, y) \implies Happy(y))$$



# Application of Description logic

- The following questions are of interest with respect to a TBox T:
  - Given a concept C , is C satisfiable ( $\langle T , \{C (x_0)\}\rangle$ has a model);
    C is a Car and an Animal is not satisfiable
  - Given two concepts C and D, is C subsumed by D ($T \models C \sqsubseteq D$);
    Quadcopters are helicopters
  - Given two concepts C and D, are C and D equivalent ($T \models C \equiv D$);
    Car and Automobile are equivalent
  - Given two concepts C and D, are C and D disjoint ($T \models C \sqcap D \sqsubseteq \bot$);
    Cars and Animals are disjoint
- The following problems are of interest with respect to knowledge bases K= $\langle A,T\rangle$
  - Is K consistent (K has a model);
  - Given a concept C and an individual *a*, does K entail *C (a) (K ⊨ C (a))*
    Is *a* a robot?
  - Given a concept C, find all individuals a such that K entails C (a).
    Give me all the victims in the environment



# Extensions to ALC

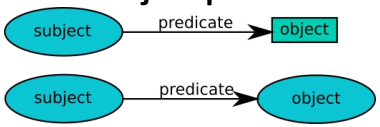
- Role hierarchies (H)
  $hasFather \sqsubseteq hasParent$
- Complex role hierarchies (R)
  $hasParent \circ hasBrother \sqsubseteq hasUncle$
- Transitivity
  $hasAncestor \circ hasAncestor \sqsubseteq hasAncestor$
- Inverse role (I)
  $hasParent \equiv hasChild^{-}$
- Cardinality restrictions (N)
  $Mammal \sqsubseteq \leq 2 hasParent$
- Qualified cardinality restrictions (Q)
  $RichPeople \sqsubseteq \geq 2owns.House$
- Closed classes (O)
  $Days \sqsubseteq \{monday, tuesday, \ldots, sunday\}$
- ...



# RDF and the Semantic Web

# RDF

- RDF stands for *Resource Description Framework*
- It is a framework for representing information in the Web
- Basic building block: **subject-predicate-object** triple




  - subject/predicate/object are entities called *resources*
  - subject-predicate-object triple is called *statement*



# Resources as URI

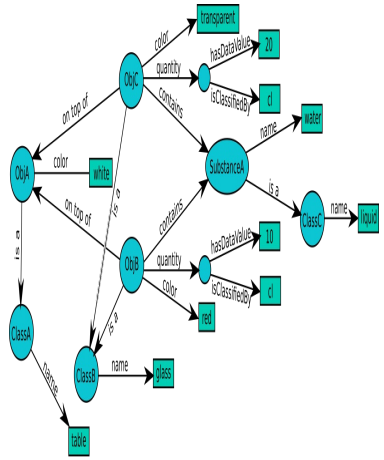
- Resources are represented as URI
  - May or may not correspond to a valid website/ document
  - Way of generating unique identifier



# Turtle

```
@base: <http://www.ida.liu.se/~TDDE05>
@prefix relations: <relations>
@prefix properties: <http://www.ida.liu.se/~TDDE05/properties>
@prefix substances: <http://www.substances.org/>
@prefix classes: <http://www.classes.org/>
@prefix dul: <http://www.loa-cnr.it/ontologies/DUL.owl#>
@prefix un:  <http://www.w3.org/2007/ont/unit#>

<ObjA> a classes:ClassA;
       properties:color "white" .
<ObjB> a classes:ClassB;
       relations:on_top_of <ObjA> ;
       properties:color "red" ;
       properties:quantity [ dul:hasDataValue 10 ; dul:isClassifiedBy   un:cl ] ;
       relations:contains substances:SubstanceA .
<ObjC> a classes:ClassB;
       relations:on_top_of <ObjA> ;
       properties:color "transparent" ;
       properties:quantity [ dul:hasDataValue 20 ; dul:isClassifiedBy   un:cl ] ;
       relations:contains substances:SubstanceA .
```

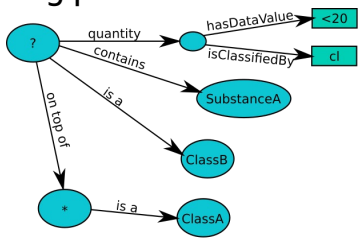





# SPARQL

# Graph matching

- How to access part of the knowledge?
  - We want to query for all glasses on top of a table with less than 15cl of water
- A graph matching problem






# SPARQL

- Syntax very similar to
  - `SELECT <variables> FROM <sources> WHERE <graph constraints>`
  - SELECT specifies the variables and values
  - FROM specifies the data source
  - WHERE imposes constraints on solution
- Graph is expressed very similarly than with
- Resources can be URI (<>), Variables (?var) or BlankNode (_:name)
  - Query everything:
    `SELECT ?x ?y ?z WHERE { ?x ?y ?z }`



# SPARQL - Examples (1/2)

- Select everything which has a color:

```
PREFIX properties: <http://www.ida.liu.se/~TDDE05/properties>
SELECT ?x ?z WHERE { ?x properties:color ?z }
```

- Select everything which has the color red:

```
PREFIX properties: <http://www.ida.liu.se/~TDDE05/properties>
SELECT ?x WHERE { ?x properties:color "red" }
PREFIX properties: <http://www.ida.liu.se/~TDDE05/properties>
SELECT ?x WHERE { ?x properties:color ?z FILTER(?z = "red") }
```

- Select everything that has less than 15cl

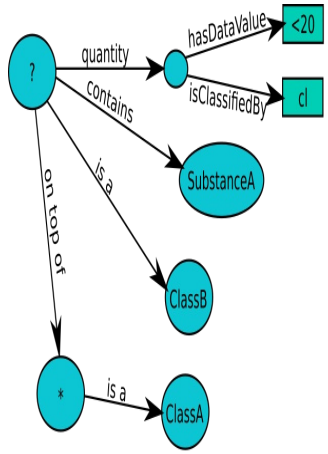
```
PREFIX properties: <http://www.ida.liu.se/~TDDE05/properties>
PREFIX dul: <http://www.loa-cnr.it/ontologies/DUL.owl#>
PREFIX un:  <http://www.w3.org/2007/ont/unit#>
SELECT ?x WHERE { ?x properties:quantity _:blankNode1 .
                  _:blankNode1 dul:hasDataValue ?z .
                  _:blankNode1 dul:isClassifiedBy un:cl .
                  FILTER(?z < 15) }
```



# SPARQL - Examples (2/2)

```
BASE: <http://www.ida.liu.se/~TDDE05>
PREFIX relations: <relations>
PREFIX properties: <http://www.ida.liu.se/~TDDE05/
properties>
PREFIX substances: <http://www.substances.org/>
PREFIX classes: <http://www.classes.org/>
PREFIX dul: <http://www.loa-cnr.it/ontologies/
DUL.owl#>
PREFIX un:  <http://www.w3.org/2007/ont/unit#>

SELECT ?glass WHERE {
  ?glass is classes:ClassB ;
         relations:contains substances:SubstanceA ;
         relations:on_top_of ?table ;
         properties:quantity [
           dul:hasDataValue ?quantity ;
           dul:isClassifiedBy un:cl ] .
  FILTER( ?quantity < 15 )
  ?table is classes:ClassA .
}
```



# RDF Limitation

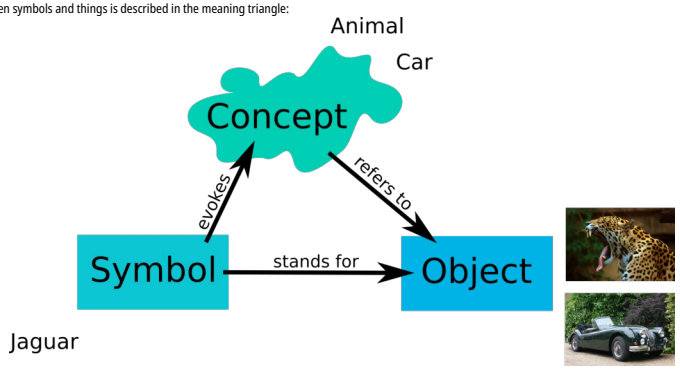
- RDF, Turtle, XML is syntax
  - It allows specification of a graph and values
  - It allows definition of explicit knowledge
- Need structure!
- Definition of implicit knowledge
- Standardization
  - <cost> £10</cost> vs <price>£10</price>
  - What does friend(ann, jane) means?
  - ...



# Ontologies

# Triangle of reference

- Humans require *words* (symbols) to communicate, they map to concept and to specific instance
- Relations between symbols and things is described in the meaning triangle:





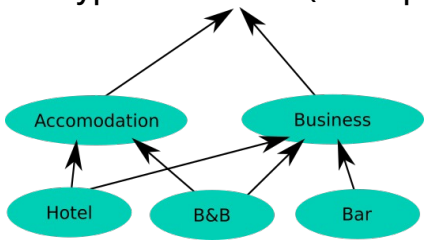

# What is an ontology? (1/2)

A common language is defined by:

- **Syntax:** common symbols and concepts
- **Semantics:** agreement about their meanings
- **Taxonomy:** classification of concepts
- **Thesauri:** associations and relations of concepts
- **Ontolgies:** rules and knowledge about whcih relations are allowed and make sense

# What is an ontology? (2/2)

- A classification of types of entities (concepts/classes):




- Constraints on how they are used:

$$\forall x, x \in Hotel \Longrightarrow x \notin B\&B$$
$$\forall x, \exists y \quad x \in Hotel \Longrightarrow partof(x, y) \wedge y \in Bar$$



# OWL



# OWL

- OWL stands for Web Ontology Language
- Expresses ontology as RDF
- Based on Description Logic (SHOIN/ SROIQ)



# Description Logic Syntax

- Axiom (ABox)

:mary a :Person

$$Person(mary)$$

:john :hasWife :mary

$$hasWife(john, mary)$$

- Concept (TBox)

:Woman owl:SubClassOf :Person

$$Woman \sqsubseteq Person$$

:Person ownl:EquivalentClass :HumanBeing

$$Person \equiv HumanBeing$$

:hasWife owl:SubPropertyOf :hasSpouse

$$hasWife \sqsubseteq hasSpouse$$

## What can we do with an ontology?

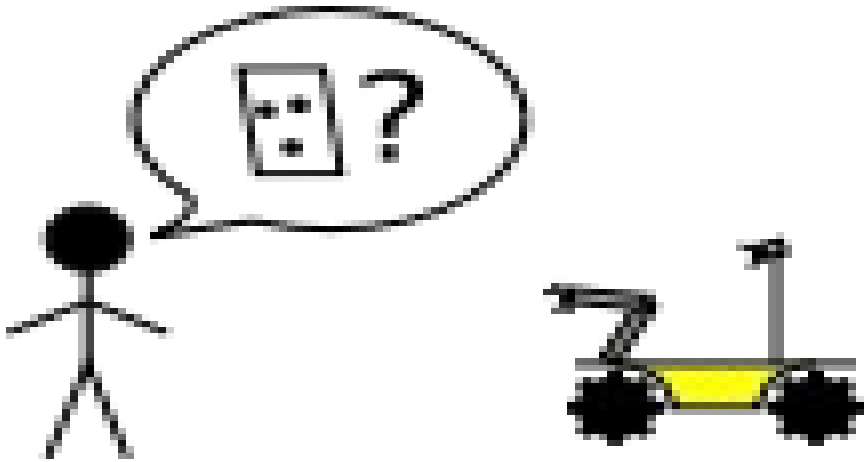
- Define terms and concepts
- Check the validity of our data
- Infer new facts
  - For instance, to ask questions such as give me the object that contain water



# Ontologies for Robotics



## How to get help with making pancakes? (1/2)

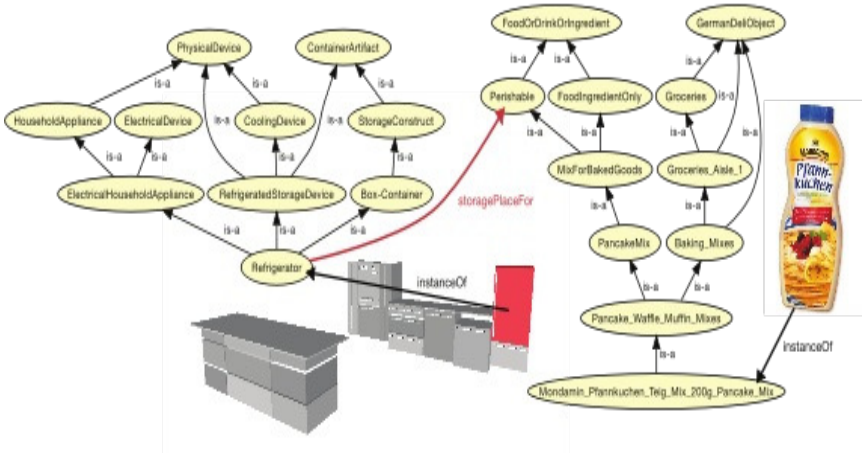





## How to get help with making pancakes? (2/2)

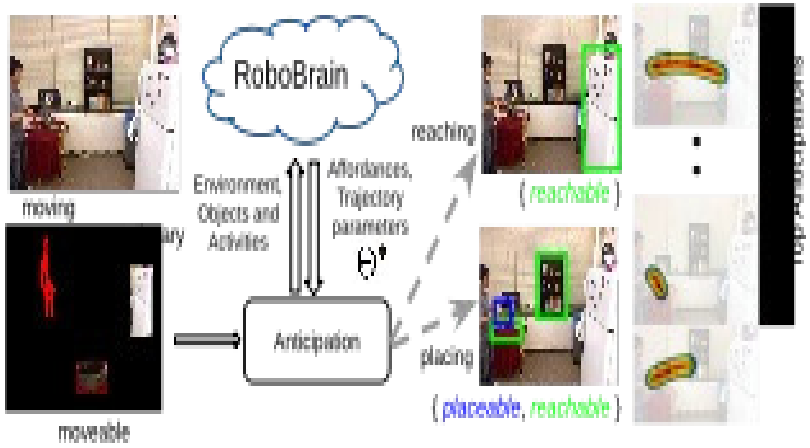
- Where is located the object?
- How to grab the object?
- …

# Where is located a cooking ingredient?






# Predict the future






# Knowledge database

- Grounded in Description Logic
- Represented as a RDF graph
- Queried with SPARQL
- OWL Ontologies



# Conclusion

- Storing symbolic data in a graph
- Querying the graph
- Structuring the knowledge