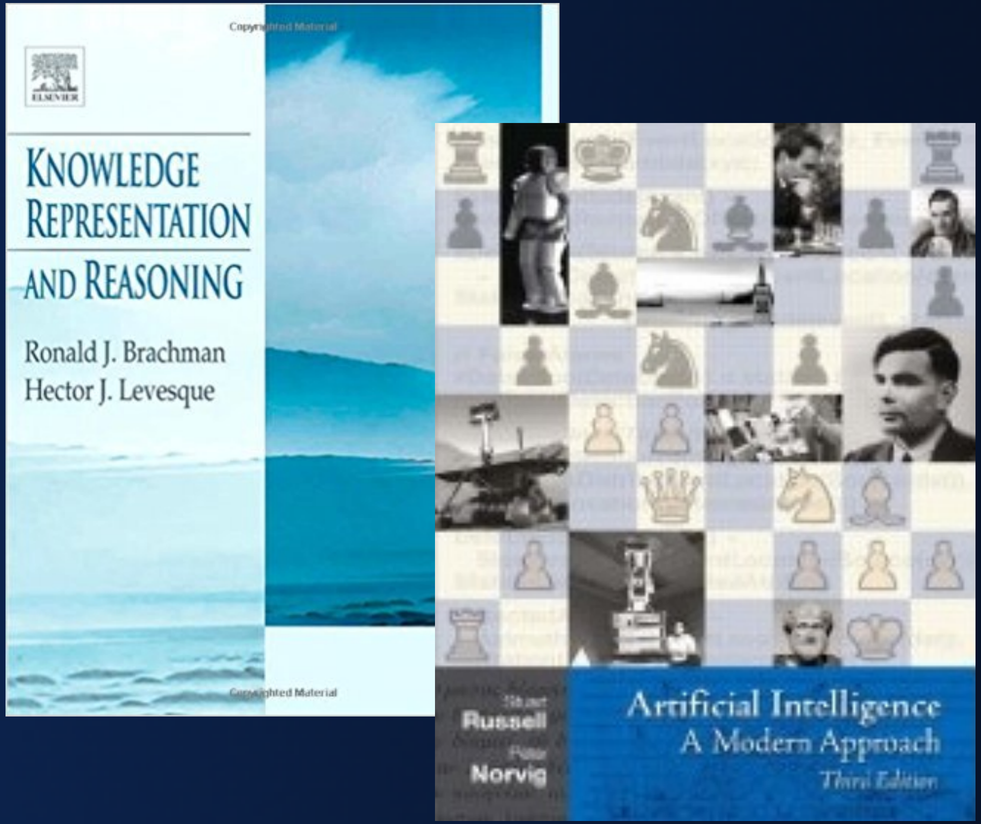



# AI Fundamentals: Knowledge Representation and Reasoning

*Maria Simi*

# Knowledge engineering and Ontology engineering

LESSON 2: SITUATION CALCULUS – EVENT AND PROCESS CALCULUS

# Knowledge engineering & Ontological engineering

We start with representation. It is possible to discuss representation issues at two levels.

**Knowledge engineering** is the activity to formalize a **specific application domain**. It involves decisions about:

1. What are the relevant, facts, objects relations …
2. Which is the right level of abstraction
3. What are the queries to the KB (inferences)

**Ontology engineering** seeks to build a **general-purpose** ontology which should be applicable in any special-purpose domain (with the addition of domain-specific axioms). For example:

Objects and categories, composite objects, bunches, substances, measurements, actions and change, events, temporal intervals … [AIMA cap. 12]

Defaults (non monotonic reasoning), knowledge and beliefs

In any non trivial domain, **different areas of knowledge** must be combined.

# Knowledge engineering: a simple example

Before implementing, need to understand clearly, like in software engineering

- what is to be computed?
- what kind of knowledge?
- why and where inference is necessary?

Task: KB with appropriate knowledge and entailments

- Assuming FOL as representation language, the kinds of objects that will be important to the agent, their properties, and the relationships among them
- the vocabulary and relations among terms.
- what facts to represent

Example domain: **soap-opera world** (about human relationships and behavior) [KRR, Ch. 3]

- people and their relationships, places, companies, marriages, divorces, “hanky-panky”, deaths, kidnappings, crimes, money ...

# Ontology and vocabulary

We need to define names for individuals and domain-dependent predicates and functions.

**Named individuals**

- *john, sleezyTown, faultyInsuranceCorp, fic, johnQsmith, …*

**Basic types**

- *Person, Place, Man, Woman, …*

**Attributes**

- *Rich, Beautiful, Unscrupulous, …*

**Relationships**

- *LivesAt, MarriedTo, DaughterOf, HadAnAffairWith, Blackmails, …*

**Functions**

- *fatherOf, ceoOf, bestFriendOf, …*

# Basic facts: atomic sentences

**Type facts**

- $Man(john)$,
- $Woman(jane)$,
- $Company(faultyInsuranceCorp)$

**Property facts**

- $Rich(john)$,
- $\neg HappilyMarried(jim)$,
- $WorksFor(jim, fic)$

**Equality facts**

- $john = ceoOf(fic)$,
- $fic = faultyInsuranceCorp$
- $bestFriendOf(jim) = john$

So far, like a **simple database** (can store in a table)

# Complex facts

**Universal assertions** (abbreviations)

- $\forall y\,[Woman(y) \land y \neq jane \Rightarrow Loves(y, john)]$ ”All the women, maybe not Jane, love John”
- $\forall y\,[Rich(y) \land Man(y) \Rightarrow Loves(y, jane)]$ “All the rich men love Jane.”
- $\forall x\,\forall y[Loves(x, y) \Rightarrow \neg Blackmails(x, y)]$ “Nobody blackmails a loved one”

**Incomplete knowledge** (relates to expressivity)

- $Loves(jane, john) \lor Loves(jane, jim)$ which?
- $\exists x\,[Adult(x) \land Blackmails(x, john)]$ who?

**Closure axioms**

- $\forall x\,[Lawyer(\mathrm{x}) \Rightarrow x{=}jane \lor x{=}john \lor x{=}jim]$
- $\forall \underline{x}\, \forall \underline{y}[MarriedTo(x, y) \Rightarrow\ (x = ethel \land \mathrm{y} = fred)\ \ldots]$
- $\forall \mathrm{x}\,[\ x{=}fic \lor x{=}jane \lor x{=}john \lor x{=}jim \ldots]$

also useful to have $jane \neq john \ldots$

# Terminological facts

General relationships among predicates. For example:

- disjoint $\forall x\, [Man(x) \Rightarrow \neg Woman(x)]$
- subtype $\forall x\, [Senator(x) \Rightarrow Legislator(x)]$
- exhaustive $\forall x\, [Adult(x) \Rightarrow Man(x) \lor Woman(x)]$
- symmetry $\forall x \forall y\, [MarriedTo(x, y) \Rightarrow MarriedTo(y, x)]$
- inverse $\forall x \forall y\, [ChildOf(x, y) \Rightarrow ParentOf(y, x)]$
- type restriction $\forall x \forall y\, [MarriedTo(x, y) \Rightarrow Person(x) \land Person(y)]$
- definitions $\forall x\, [RichMan(\mathrm{x}) \Leftrightarrow Rich(\mathrm{x}) \land Man(\mathrm{x})]$

Usually universally quantified conditionals or biconditionals

# Entailment -1

*Is there a company whose CEO loves Jane*?

$$\mathrm{KB} \vDash \exists x\, [Company(x) \wedge Loves(ceoOf\,(x), jane)] \;??$$

Suppose $\mathrm{KB}$ is true,

then $Rich(john)$, $Man(john)$, $\forall y\, [Rich(y) \wedge Man(y) \Rightarrow Loves(y, jane)]$ are true

so $Loves(john, jane)$ Also $john = ceoOf(fic)$

so $Loves(ceoOf(fic), jane)$

Finally $Company(faultyInsuranceCorp)$, and $fic = faultyInsuranceCorp$,

so $Company(fic)$

Thus, $Company(fic) \wedge Loves(ceoOf(x), jane)$

so $\exists x\, [Company(x) \wedge Loves(ceoOf(x), jane)$

Can extract identity of company from this proof

# Entailment - 2

If no man is blackmailing John, then is he being blackmailed by somebody he loves?

$$\text{KB} \models \forall x\,[Man(x) \Rightarrow \neg Blackmails(x, john)] \Rightarrow$$
$$\exists y\,[Loves(john, y) \;\wedge Blackmails(y, john)]?$$

Show: $\text{KB} \cup \forall x\,[Man(x) \Rightarrow \neg Blackmails(x, john)] \models$
$$\exists y\,[Loves(john, y) \;\wedge Blackmails(y, john)]$$

…

$$Loves(john, jane) \wedge Blackmails(jane, john)]$$

# Abstract individuals and reification

Sometimes useful to reduce *n*-ary predicates to 1-place predicates and 1-place functions

- involves reifying properties, creating new individuals
- typical of description logics / frame languages (later)

Flexibility in terms of arity:

$Purchases(john, sears, bike)$ or

$Purchases(john, sears, bike, feb14)$ or

$Purchases(john, sears, bike, feb14, \$100)$

Instead: introduce individuals for purchase objects and functions for roles (**reification**)

$Purchase(\underline{p23}) \wedge agent(p23) = john \wedge object(p23) = bike \wedge source(p23) = sears \wedge amount(p23) = \$200 \wedge \ldots$

allows purchase to be described at various levels of detail.

For talking about ages and money, we need to decide how to deal with **measurements**.

# Other sort of facts requiring FOL extensions

**Statistical / probabilistic facts**

- *Half of the companies are located on the East Side.*
- *Most of the employees are restless.*
- *Almost none of the employees are completely trustworthy,*

**Default / prototypical facts**

- *Company presidents typically have secretaries intercepting their phone calls.*
- *Cars have four wheels.*
- *Companies generally do not allow employees that work together to be married.*

**Intentional facts**

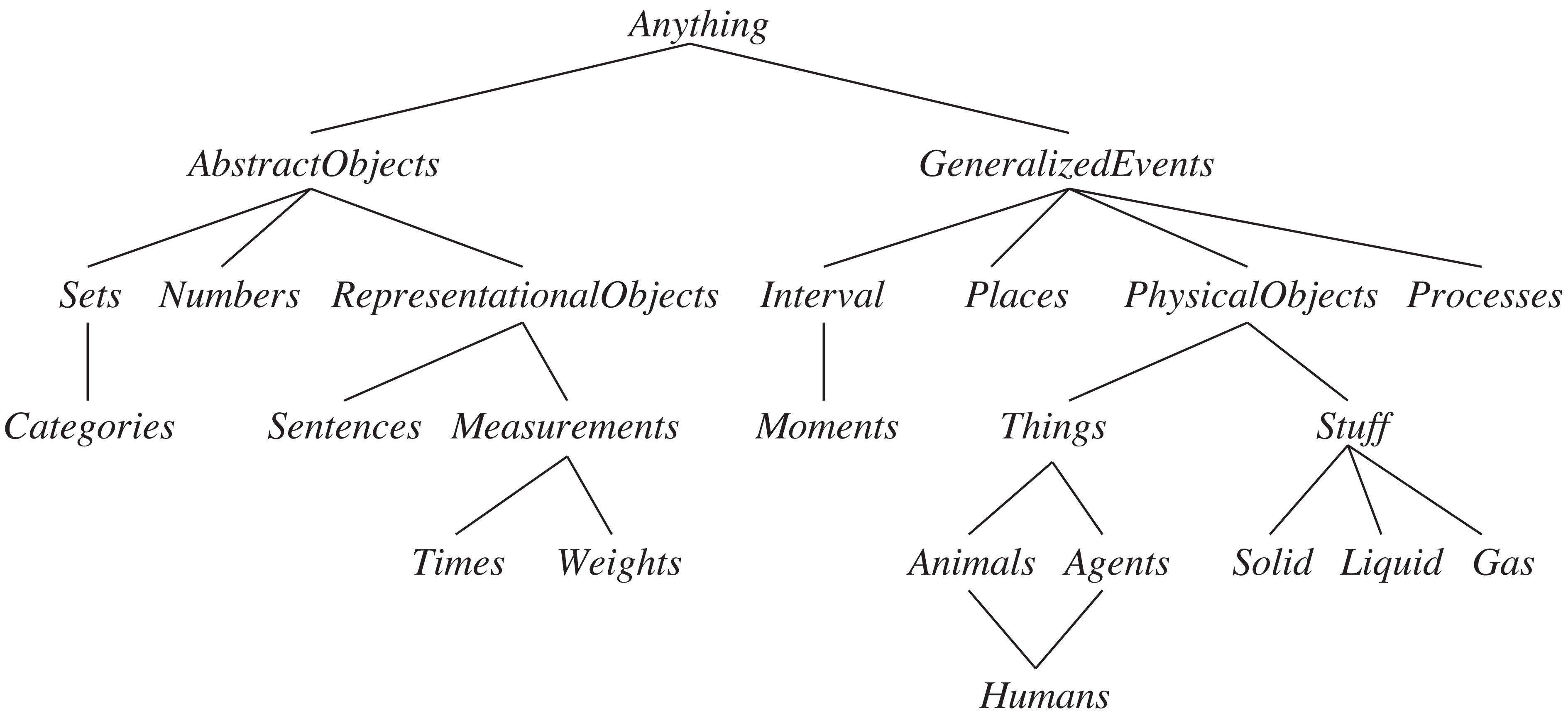
- *John believes that Henry is trying to blackmail him.*
- *Jane does not want Jim to think that she loves John.*

Others ...

# Representing common sense [AIMA cap 12]

- The use of KR languages and logic in A.I. is representing "*common sense*" knowledge about the world, rather than mathematics or properties of programs.
- Common sense knowledge is difficult since it comes in different varieties. It requires formalisms able to represent *actions, events*, *time*, *physical objects*, *beliefs* … categories that occur in many different domains.
- In this lecture we will explore FOL as a tool to formalize different kinds of knowledge.
- A lot of intersections with philosophical logic, but in A.I. the emphasis is also on reasoning and its complexity.

# General/upper ontology




A general ontology organizes everything in the world into a hierarchy of categories.

# Properties of general-purpose ontologies

- A general-purpose ontology should be applicable in any special-purpose domain (with the addition of domain-specific axioms).
- In any non trivial domain, different areas of knowledge must be combined, because reasoning and problem solving could involve several areas simultaneously.
- Difficult to construct one best ontology. “Every ontology is a treaty—a social agreement—among people with some common interest in sharing.”
- Several attempts:
  - CYC (Lenat and Guha, 1990); OpenMind (MIT project); DBpedia (Bizer et al., 2007)
  - Parsing text documents and extracting information from them (e.g. TextRunner ...)
  - The ontologies of the semantic web [see **Semantic web** course]

# Categories and objects

Much reasoning takes place at the level of categories: we can infer category membership from the perceived properties of an object, and then uses category information to derive specific properties of the object.

There are two choices for representing categories in first-order logic:

1. Predicates, categories are unary predicates, that we assert of individuals:

   $Sport(tennis)$

2. Objects: categories are objects that we talk about (**reification**)

   $tennis \in Sports$

   $WinterSports \subseteq Sports$

This way we can organize categories in **taxonomies** (like in natural sciences), define disjoint categories, partitions … and use specialized inference mechanisms. Problems with **natural kinds**, which do not admit logical definitions.

# Composite objects: *part-of*

We use the general $PartOf$ relation to say that one thing is part of another.

Composite objects can be seen as **part-of hierarchies**, similar to the $Subset$ hierarchy. These are called **mereological** hierarchies.

$$PartOf(nose, face)$$
$$PartOf(Bucharest, Romania)$$
$$PartOf(Romania, EasternEurope)$$
$$PartOf(EasternEurope, Europe)$$
$$PartOf(Europe, Earth)$$

The $PartOf$ relation is transitive and reflexive:

$$PartOf(x, y) \land PartOf(y, z) \Rightarrow PartOf(x, z)$$
$$PartOf(x, x)$$

# Composite objects: structural relations

Structural relations among parts.

For example, *a biped has two legs attached to a body*:

$$Biped(a) \Rightarrow \exists\, l_1, l_2, b$$

$$Leg(l_1) \land Leg(l2) \land Body(b) \land$$

$$PartOf\,(l_1, \mathrm{a}) \land PartOf\,(l_2, \mathrm{a}) \land PartOf\,(b, a) \land$$

$$Attached(l_1, b) \land Attached(l_2, b) \land$$

$$l_1 \neq l_2 \land [\forall l_3\ Leg(l_3) \land PartOf\,(l_3, a) \Rightarrow (l_3 = l_1 \lor l_3 = l_2)]$$

*esattamente due gambe!*

# Composite objects: *bunches*

Composite objects with definite parts but no particular structure.

E.g. "a bag of three apples".

$BunchOf(\{Apple_1, Apple_2, Apple_3\})$ not to be confused with the set of 3 apples

$BunchOf(Apples)$ is the composite object consisting of all apples—not to be confused with $Apples$, the category or set of all apples.

How objects, bunches, sets and categories relate?

1. $BunchOf(\{x\}) = x$
2. Each element of category $s$ is part of $BunchOf(s)$:
   $\forall x \,.\, x \in Apples \Rightarrow PartOf(x, BunchOf(Apples))$
3. $BunchOf(s)$ is the smallest object satisfying this condition.
   $\forall y\, [\forall x \;\; x \in s \Rightarrow PartOf(x, y)] \Rightarrow PartOf(BunchOf(s), y)$
   $BunchOf(s)$ must be part of any object that has all the elements of $s$ as parts

# Quantitative measures

Physical objects have height, weight, mass, cost, and so on. The values that we assign for these properties are called **measures.**

A solution is to represent measures with **units functions** that take a number as argument.

$$Length(L_1) = Inches(1.5) = Centimeters(3.81)$$

$$Centimeters(2.54 \times d) = Inches(d)$$

$$Diameter\,(Basketball_{12}) = Inches(9.5)$$

$$ListPrice(Basketball_{12}) = \$(19)$$

$$d \in Days \Rightarrow Duration(d) = Hours(24)$$

$L_1$

# Qualitative measures

The most important aspect of measures is not the particular numerical values/scale, but the fact that measures can be ordered.

For example, we might well believe that *Norvig's exercises are tougher than Russell's, and that one scores less on tougher exercises*:

$$e_1 \in Exercises \wedge e_2 \in Exercises \ \wedge Wrote(Norvig, e_1) \wedge Wrote(Russell, e_2) \Rightarrow Difficulty(e_1) > Difficulty(e_2)$$

$$e_1 \in Exercises \wedge e_2 \in Exercises \wedge Difficulty(e_1) > Difficulty(e_2) \Rightarrow ExpectedScore(e_1) < ExpectedScore(e_2)$$

To perform some sort of **qualitative inference**, often it is enough to be able to order values and to compare quantities (qualitative physics)

# Objects vs stuff

There are **countable objects**, things such as apples, holes, and theorems, and **mass objects**, such as butter, water, and energy. These are called *Stuff*.

Properties of **stuff:**

1. Any part of butter is still butter:

   $b \in \boldsymbol{Butter} \land PartOf(p, b) \Rightarrow p \in Butter$

2. Stuff has a number of **intrinsic properties** (color, high-fat content, density ...), shared by all its subparts, but no **extrinsic properties** (weight, length, shape ...). It is a **substance**.

# The situation calculus in FOL

The situation calculus is a specific ontology dealing with actions and change:

- **Situations**: snapshots of the world at a given instant of time, the result of an action.
- **Fluents**: time dependent properties
- **Actions**: performed by an agent, but also events.
- **Change**: how the world changes as a result of actions

The situation calculus is formalization in FOL of this ontology [Mc Carthy, 69]

# The blocks world

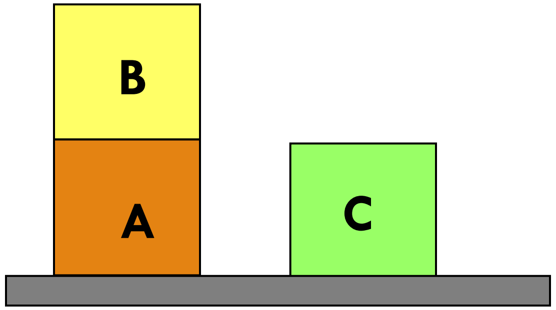
*A scenario much used in planning. The are blocks on a table and the goal is to reach a given arrangement of the blocks by stacking them on top of each other.*

**States**: arrangements of blocks on a table

**Initial state** and **goal state**: a specific arrangement of blocks

Actions:

- **move**: move block $x$ from block $y$ to block $z$, provided $x$ and $\underline{z}$ are free.
- **unstack**: move block $x$ from $y$ to the table. $x$ must be free.
- **stack**: move $x$ from the table to $y$. $y$ must be free.



# The blocks world formalization in FOL

- **Situations**: $s$, $s_0$, $s_1$, $s_2$ … and functions denoting situations
- **Fluents**: predicates or functions that vary from a situation to another:

  *On, Table, Clear … Hat*

  $On(a, b)$ becomes $On(a, b, s)$

  $Hat(a)$ becomes $Hat(a, s)$

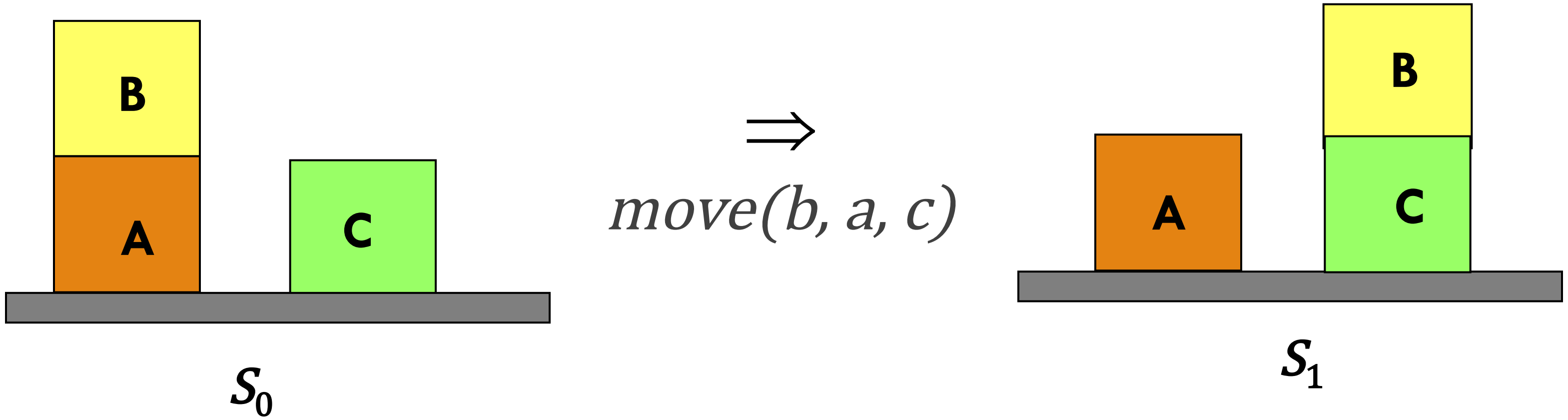
  Immutable properties are represented as before (e.g. $Block$)
- **Actions**: are modelled as functions (terms)

  $move(a, b, c)$

  is a function representing the action of moving block $A$ from $B$ to $C$.
  It is an instance of the generic operator/function $move$.

  Similarly for $unstack(a, b)$ and $stack(a, b)$.

# Situations as result of actions




- Effect of actions: function $Result: A \times S \rightarrow S$

  $s_1 = Result\,(move(b, a, c), s_0)$

  denotes the situation resulting from the action $move(b, a, c)$ executed in $s_0$.

  Then we can assert for example:

  $On(b, c, Result\,(move(b, a, c), s_0))$

# Result of a sequence of actions

Effect of a sequence of actions: $Result: [A^*] \times S \rightarrow S$

1. $Result([\,], s) = s$
2. $Result([a \mid seq], s) = Result(seq, Result(a, s))$

For example:

$Result([move(a, b, c), stack(a, b)], s_0) \equiv$
$Result\,(stack(a, b), Result\,(move(a, b, c), s_0))$

In general:

$Result([a_1, a_2, \ldots a_n], s_0) \equiv$
$Result\,(a_n, Result\,(a_{n\text{-}1}, \ldots Result\,(a_2, Result\,(a_1, s_0)) \ldots)$

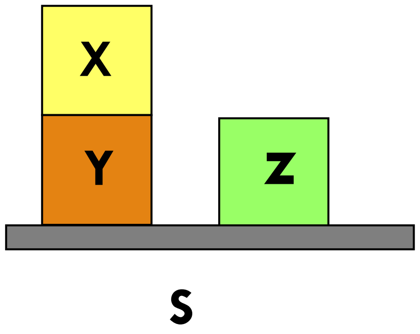
# Formalizing actions

- We need **possibility axioms** with this structure: $preconditions \Rightarrow poss$

  $On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge x \neq z \Rightarrow$

  $\qquad Poss(move(x, y, z), s)$

  Note: Variables are universally quantified.




- And **effect axioms** such as:

  $Poss(move(x, y, z), s) \Rightarrow$

  $\qquad On(x, z, Result(move(x, y, z), s)) \wedge Clear(y, Result(move(x, y, z), s))$

- This is not enough however: Is $y$ on the table in the new situation? Is $x$ free?
- We have a [big] problem: in the new situation we do not know anything about properties that were not influenced at all by the action. These are the majority!!!
- This is the **frame problem**.

# The *frame problem* and *frame axioms*.

The **frame problem** is one the most classical A.I. problems [McCarthy-Hayes, 1969]. There is an analogy with the animation world, where the problem is to distinguish *background* (the fixed part) from the *foreground* (things that change) from one frame to the other.

Let's fix that writing **frame axioms**.

*Frame axioms* for *Clear* with respect to *move*:

$$Clear(x, s) \wedge x \neq w \Rightarrow Clear(x, Result(move(y, z, w), s))$$

A block stays free unless the *move* action is putting something on it.

$$\neg Clear(x, s) \wedge x \neq z \Rightarrow \neg Clear(x, Result(move(y, z, w), s))$$

A block remains not free unless it is not freed by the action.

And similarly for each pair *fluent-action*. Too many axioms (**representational frame problem**)

# Successor-state axioms [Reiter 1991]

We can combine preconditions, effect and frame axioms to obtain a more compact representation **for each fluent** *f*. The schema is as follows:

| | |
|---|---|
| *f true after* ⇔ *preconditions and* | *preconditions* |
| [*some action made f true* or | *effect* |
| *f was true before and no action made it false*] | *frame axioms* |

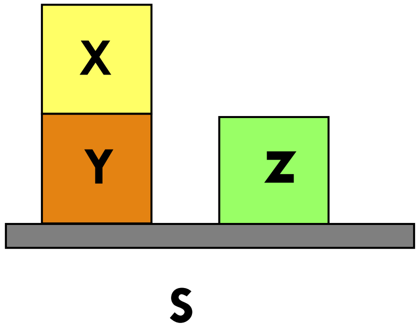
Example: *state-successor* axiom for fluent *Clear*:

$$Clear(y, Result(a, s)) \Leftrightarrow$$

$$[On(x, y, s) \land Clear(x, s) \land Clear(z, s) \land x \neq z \land a = move(x, y, z))] \lor$$

$$[On(x, y, s) \land Clear(x, s) \land (a = unstack(x, y))] \lor$$

$$[Clear(y, s) \land (a \neq move(z, w, y)) \land (a \neq stack(z, y))]$$



# Deriving successor-state axioms

**Positive** and **negative** effect axioms, stating a fluent becomes true [false].

$$+ \; On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge x \neq z \Rightarrow Clear(y, Result(move(x, y, z), s))$$

$$+ \; On(x, y, s) \wedge Clear(x, s) \Rightarrow Clear(y, Result(unstack(x, y), s))$$

$$- \; Clear(w, s) \wedge Clear(y, s) \Rightarrow \neg Clear(y, Result(move(w, x, y), s))$$

$$- \; Clear(w, s) \wedge Table(w, s) \Rightarrow \neg Clear(y, Result(stack(w, y), s))$$

Rewrite as a single formula the positive effects:

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge x \neq z \wedge a = move(x, y, z))] \vee$$
$$[On(x, y, s) \wedge Clear(x, s) \wedge (a = unstack(x, y))] \Rightarrow Clear(y, Result(a, s))$$

Assume these are the only actions producing that positive effects:

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge x \neq z \wedge a = move(x, y, z)] \vee$$
$$[On(x, y, s) \wedge Clear(x, s) \wedge (a = unstack(x, y)] \Leftrightarrow Clear(y, Result(a, s)) \qquad (1)$$

Moreover we assume $move(x, y, z)$ and $unstack(x, y)$ are different actions

# Deriving successor-state axioms (cnt.)

For negative effects

$$[Clear(w,s) \wedge Clear(y,s) \wedge a = move(w,x,y) \vee$$
$$Clear(w,s) \wedge Table(w,s) \wedge a = stack(w,y)] \Rightarrow \neg Clear(y, Result(a,s))$$

By closure (these are the only actions making Clear false, provided it was not false already):

$$\neg Clear(y,s) \vee [Clear(w,s) \wedge Clear(y,s) \wedge a = move(w,x,y)] \vee$$
$$[Clear(w,s) \wedge Table(w,s) \wedge a = stack(w,y)] \Leftrightarrow \neg Clear(y, Result(a,s))$$

Negating both members and simplifying we get frame axioms:

$$[Clear(y,s) \wedge a \neq move(w,x,y) \wedge a \neq \mathrm{unstack}(w,y)] \Leftrightarrow \qquad (2)$$
$$Clear(y, Result(a,s))$$

Putting (1) and (2) together, we obtain the successor state axiom.

# Deriving successor-state axioms in general

**Positive** and **negative** effect axioms, stating a fluent becomes true [false].

$$\mathrm{P}(\boldsymbol{x}, a, s) \Rightarrow F(\boldsymbol{x}, Result(a, s)) \qquad (1)$$

$$\mathrm{N}(\boldsymbol{x}, a, s) \Rightarrow \neg F(\boldsymbol{x}, Result(a, s)) \qquad (2)$$

Completeness assumptions, called **explanation closures**:

$$\neg F(\boldsymbol{x}, \mathrm{s}) \land [\mathrm{P}(\boldsymbol{x}, a, s) \equiv F(\boldsymbol{x}, Result(a, s))] \qquad (3)$$

$$F(\boldsymbol{x}, \mathrm{s}) \land [\neg \mathrm{N}(\boldsymbol{x}, a, s) \equiv F(\boldsymbol{x}, Result(a, s))] \qquad (4)$$

**Successor state axiom** for fluent $F$:

$$F(\boldsymbol{x}, Result(a, s)) \equiv [\neg F(\boldsymbol{x}, \mathrm{s}) \land \mathrm{P}_F(\boldsymbol{x}, a, s)] \lor [F(\boldsymbol{x}, s) \land \neg \mathrm{N}_F(\boldsymbol{x}, a, s)]$$

$F$ is true after doing $a$ iff $a$ made it true or it was true before and it was not made false by any other action.

# Related problems

The **representational frame problem** is considered to be (more or less) solved.

**Qualification problem**: in real situations it is almost impossible to list all the necessary and relevant preconditions.

$$Clear(x) \wedge Clear(y) \wedge Clear(z) \wedge y \neq z \wedge \neg Heavy(x) \wedge \neg Glued(x) \wedge \neg Hot(x) \wedge \ldots \Rightarrow move(x, y, z)$$

**Ramification problem**: among derived propertied which ones persist and which ones change?

- Objects on a table are in the room where the table is. If we move the table from one room to another, objects on the table must also change their location. Frame axioms could make the objects make the old location persist.

# Uses of situation calculus

**Planning:** finding a sequence of actions to reach a certain goal state.

**Projection:** Given a sequence of actions and some initial situation, determine what it would be true in the resulting situation.

Given $\Phi(s)$ determine whether $KB \vDash \Phi(Result(\boldsymbol{a}, s_0))$ where $\boldsymbol{a} = [a_1, \ldots, a_n]$

**Legality test:** Checking whether a given sequence of actions $[a_1, \ldots, a_n]$ can be performed starting from an initial situation.

$KB \vDash Poss(a_i, Result([a_1, \ldots, a_{i-1}], s_0))$ for each $i$ *such that* $1 \leq i \leq n$

For example:

$Result(pickup(b_2), Result(pickup(b_1), s_0))$

Would not be a legal situation, given that the robot can hold only one object.

# Nonmonotonic approach to the frame problem

What we would need is the ability to formalize a notion of **persistence**:

*"in the absence of information to the contrary (by default) things remain as they were"*.

Unfortunately this leads out of classical logic. Next lecture.

The **closure assumption** we used is already an *ad hoc* form of completion and we will see more of this strategy in nonmonotonic reasoning.

In planning we end up using other languages that make stronger assumptions and are more limited in their expressivity.

# Limits of situation calculus

Situation calculus is limited in its applicability:

1. Single agent
2. Actions are discrete and instantaneous (no duration in time)
3. Actions happen one at a time: no concurrency, no simultaneous actions
4. Only primitive actions: no way to combine actions (conditionals, iterations …)

To handle such cases we introduce an alternative formalism known as **event calculus**, which is based on **events**, **points in time**, **intervals** rather than situations.

# Event calculus: reification of fluents

Event calculus **reifies** *fluents* and *events*.

The fluent is an object (represented by a function).

$At(Shankar, Berkeley)$

This is a term and does not by itself say anything about whether it is true.

To assert that a **fluent is true** at some point in time $t$ we use the predicate $T$:

$T(At(Shankar, Berkeley), t)$

# Event calculus: reification of events

Events are described as instances of **event categories**.

The event $E_1$ of Shankar **flying** from San Francisco to Washington, D.C. is described as

$$E_1 \in Flyings \land Flyer(E_1 , Shankar) \land Origin(E_1 , SF) \land Destination(E_1, DC)$$

By reifying events we make it possible to add any amount of arbitrary information about them. For example, we can say that *Shankar's flight was bumpy* with $Bumpy(E_1)$.

$E_1 \in Flyings(Shankar , \underline{\mathrm{SF}}, DC)$ as an alternative

# Event calculus: intervals

Time intervals are a pair of times (*start, end*):

- $i = (t_1, t_2)$ is the time interval that starts at $t_1$ and ends at $t_2$.
- $Happens(E_1, i)$ to say that the event $E_1$ took place over the time interval $i$
- Same thing in functional form with $Extent(E_1) = i$.

The complete set of predicates for one version of the event calculus is:

| | |
|---|---|
| $\mathrm{T}(f, t)$ | Fluent $f$ is true at time $t$ |
| $Happens(e, i)$ | Event $e$ happens over the time interval $i$ |
| $Initiates(e, f, t)$ | Event $e$ causes fluent $f$ to start to hold at time $t$ |
| $Terminates(e, f, t)$ | Event $e$ causes fluent $f$ to cease to hold at time $t$ |
| $Clipped(f, i)$ | Fluent $f$ ceases to be true at some point during time interval $i$ |
| $Restored(f, i)$ | Fluent $f$ becomes true sometime during time interval $i$ |

# Event calculus: properties

A fluent holds at a point in time if the fluent was initiated by an event at some time in the past and was not made false (clipped) by an intervening event. Formally:

$$Happens(e, (t_1, t_2)) \land Initiates(e, f, t_1) \land \neg Clipped(f, (t_1, t)) \land t_1 < t \Rightarrow T(f, t)$$

A fluent does not hold at a point in time if the fluent was terminated by an event at some time in the past and was not restored by an event occurring at a later time. Formally:

$$Happens(e, (t_1, t_2)) \land Terminates(e, f, t_1) \land \neg Restored(f, (t_1, t)) \land t_1 < t \Rightarrow \neg T(f, t)$$

where *Clipped* and *Restored* are defined by

$$Clipped(f, (t_1, t_2)) \Leftrightarrow \exists e, t, t_3\ Happens(e, (t, t_3)) \land t_1 \leq t < t_2 \land Terminates(e, f, t)$$

$$Restored(f, (t_1, t_2)) \Leftrightarrow \exists e, t, t_3\ Happens(e, (t, t_3)) \land t_1 \leq t < t_2 \land Initiates(e, f, t)$$

A fluent holds over an interval if it holds on every point within the interval:

$$T(f, (t_1, t_2)) \Leftrightarrow [\forall\, t\, (t_1 \leq t < t_2) \Rightarrow T(\mathrm{f}, t)]$$

# Actions in the event calculus

Fluents and actions are related with domain-specific axioms that are similar to successor-state axioms.

For example, in the Wumpus world we can say that "*the only way to use up an arrow is to shoot it*", assuming the agent has an arrow in the initial situation:

$$Initiates(e, HaveArrow(a), t) \Leftrightarrow e = Start$$

$$Terminates(e, HaveArrow(a), \mathrm{t}) \Leftrightarrow e \in Shootings(a)$$

where *Start* denotes a distinguished event, used to describe what is true in the initial state

We can extend event calculus to make it possible to represent simultaneous events, continuous events and so on …

# Processes

**Processes** or **liquid events** are events with the property that if they happen over an interval also happen over any subinterval:

$$(e \in Processes) \land Happens\,(e, (t_1, t_4)) \land (t_1 < t_2 < t_3 < t_4) \Rightarrow Happens(e, (t_2, t_3))$$

The distinction between **liquid** and **nonliquid events** is analogous to the difference between substances, or stuff, and individual objects, or things.

For example $e \in Flyings$ is a liquid event: any small interval within a flight is still a flying event. Instead, a subevent of a trip from Milan to Rome has a different nature (perhaps a trip from Milan to Bologna).

# Time intervals

We can consider two kinds of time intervals:

1. Moments, zero duration $\quad i \in Moments \Leftrightarrow Duration(i) = Seconds(0)$
2. Extended intervals

More vocabulary:

$Time(x)$: points in a time scale, giving us absolute times in seconds

$Begin(i)$, $End(i)$: the earliest and latest moments in an interval

$Duration(i)$: the duration of an interval

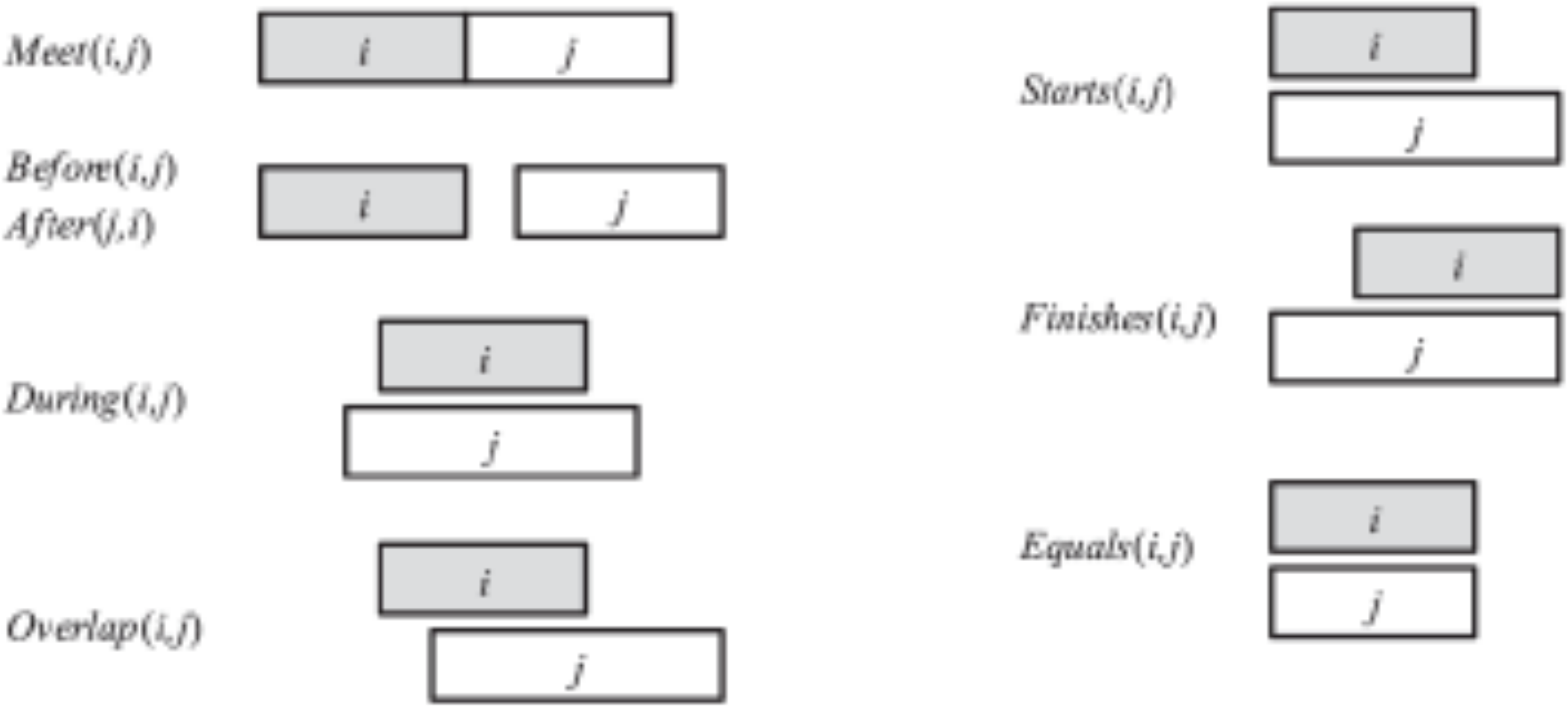
Property: $Interval\,(i) \Rightarrow Duration(i) = (Time(End(i)) - Time(Begin(i)))$

Examples:

$Time(Begin(AD2001)) = Seconds(3187324800) = Date(0, 0, 0, 1, \text{Jan}, 2001)$

$Date(0, 20, 21, 24, 1, 1995) = Seconds(3000000000)$

# Interval relations [Allen 1983]



# Time interval relations

Complete set of interval relations, proposed by Allen (1983):

$$
\begin{array}{lll}
Meet\,(\underline{i}, j) & \Leftrightarrow & End(i)=Begin(j) \\
\text{Before}\,(\underline{i}, j) & \Leftrightarrow & End\,(i) < Begin(j) \\
After\,(\underline{j}, i) & \Leftrightarrow & Before\,(\underline{i}, j) \\
During\,(\underline{i}, j) & \Leftrightarrow & Begin(j) < Begin(i) < End\,(i) < End\,(j) \\
Overlap\,(\underline{i}, j) & \Leftrightarrow & Begin(i) < Begin(j) < End\,(i) < End\,(j) \\
Begins\,(\underline{i}, j) & \Leftrightarrow & Begin(i) = Begin(j) \\
Finishes(\underline{i}, j) & \Leftrightarrow & End(i) = End(j) \\
Equals\,(\underline{i}, j) & \Leftrightarrow & Begin(i) = Begin(j) \wedge End\,(i) = End\,(j)
\end{array}
$$

Examples:

$$Meets(ReignOf\,(GeorgeVI\,), ReignOf\,(ElizabethII))$$

$$Overlap(Fifties, ReignOf\,(Elvis))$$

$$Begin(Fifties) = Begin(AD1950)$$

$$End(Fifties) = End(AD1959)$$

# Physical objects as generalized events

Physical objects, when their properties change in time, are better represented as events with a duration.

Example: $USA$ and $President(USA)$ have different properties in different periods.

$Population(USA)$, or identity of $President(USA)$ in 1789.

Proposed solution: President(USA) denotes a single object that consists of different people at different times.

$T(Equals(President(USA), GeorgeWashington), \ AD1790)$

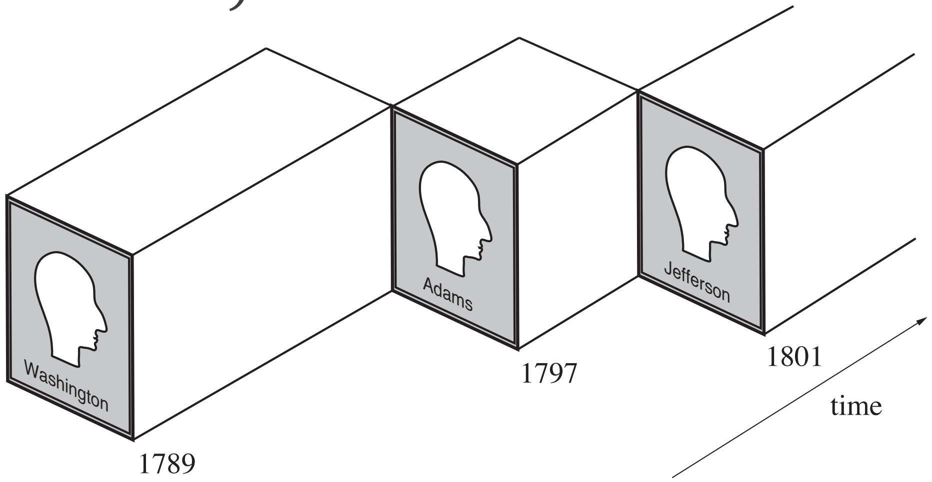
Why not $President(USA, t)$?

Not consistent with the ontology.

Why $Equals$ and not '='?

A predicate as argument of another predicate

Is not allowed by FOL



# Conclusions

- By using FOL, we discussed several representational problems, that may occur in different application domains.
- The **frame problem** is maybe the most serious one, if you want to reason about a changing world and do some KB-based planning. We will see later, how this difficulty leads to more practical approaches.
- We anticipated some of the limits of FOL, shared by all classical logics, in expressing *defaults* and *persistence*, that lead us to consider alternatives to classical logic.
- We did not talk about **mental states,** because these will be tackled in a separate lecture.

# Your turn

Knowledge engineering in another domain using FOL:

- ✓The electronic circuits domain (AIMA cap 8.4)
- ✓The Internet shopping world (AIMA cap 12.7)

Discuss some general ontological problem.

# References

[AIMA] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (3rd edition). Pearson Education 2010 (cap 4, cap 12).

[KRR] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2004. (Cap. )