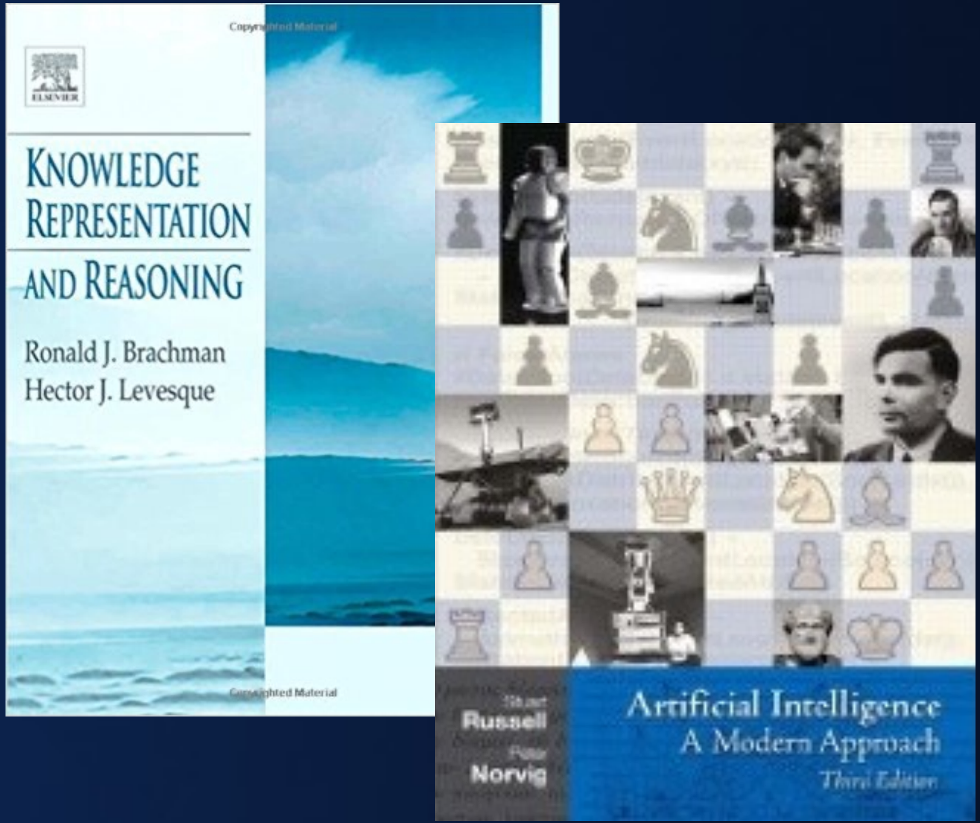


AI Fundamentals: Knowledge Representation and Reasoning

Maria Simi

Knowledge engineering and Ontology engineering

LESSON 2: SITUATION CALCULUS – EVENT CALCULUS

Knowledge engineering & Ontological engineering

It is possible to discuss representation issues at two levels.

Knowledge engineering is the activity to formalize a **specific problem or task 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 **general-purpose** ontologies which can be reused in any special-purpose domain (with the addition of domain-specific axioms). For example:

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

Defaults (non monotonic reasoning), knowledge and beliefs (will be dealt with later).

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, deaths, kidnappings, crimes, money ...

Task ontology and vocabulary (signature)

In FOL we need to define names for individuals and domain-dependent predicates and functions.

Named individuals

- *John, SleezyTown, FaultyInsuranceCorp, Fic, JohnQsmith, …*

Basic types/categories

- *Person, Place, Man, Woman, …*

Attributes

- *Rich, Beautiful, Unscrupulous, …*

Relationships

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

Functions

- *FatherOf, CeoOf, BestFriendOf, …*

Basic facts: atomic sentences

Type/category facts

- $Man(John)$,
- $Woman(Jane)$,
- $Company(FaultyInsuranceCorp)$

Properties and relations

- $Rich(John)$,
- $\neg HappilyMarried(Jim)$
- $WorksFor(Jim, Fic)$

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)

- $\forall y\,[Woman(y) \wedge y \neq Jane \Rightarrow Loves(y, John)]$ All the women, excluding Jane, love John
- $\forall y\,[Rich(y) \wedge Man(y) \Rightarrow Loves(y, Jane)]$ 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)

- $Loves(Jane, John) \vee Loves(Jane, Jim)$ which?
- $\exists x\,[Adult(x) \wedge Blackmails(x, John)]$ who?

Closure axioms

- $\forall x\,[Lawyer(x) \Rightarrow x{=}Jane \vee x{=}John \vee x{=}Jim]$ Jane, John and Jim are the only lawers
- $\forall \underline{x}\, \forall \underline{y}[MarriedTo(x, y) \Rightarrow\ (x = Ethel \wedge \mathrm{y} = Fred)\ \ldots]$ the only married people are …
- $\forall \mathrm{x}\ [\mathrm{x}{=}Fic \vee x{=}Jane \vee x{=}John \vee x{=}Jim \ldots]$ the only individuals are …

also useful to have $Jane \neq John$ … It is not taken for granted in FOL

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 $\forall x\,[Adult(x) \Rightarrow Man(x) \vee Woman(x)]$
- 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 $\forall x \forall y\,[MarriedTo(x, y) \Rightarrow Person(x) \wedge Person(y)]$
- definitions $\forall x\,[RichMan(\mathrm{x}) \Leftrightarrow Rich(\mathrm{x}) \wedge Man(\mathrm{x})]$

Usually universally quantified conditionals or biconditionals

Entailment -1

Is there a company whose CEO loves Jane?

$KB \models \exists x\, [Company(x) \wedge Loves(CeoOf(x), Jane)]$??

Suppose 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, $Company(Fic) \wedge Loves(CeoOf(Fic), Jane)$

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?

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

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

Remember:
$$\exists x\, [Adult(x) \wedge Blackmails(x, John)]$$
$$\forall x\, [Adult(x) \Rightarrow Man(x) \vee Woman(x)]$$
$$\forall x \forall y [Loves(x, y) \Rightarrow \neg Blackmails(x, y)]$$
$$\forall y\, [Woman(y) \wedge y \neq Jane \Rightarrow Loves(y, John)]$$
$$\ldots$$
$$Loves(John, Jane) \wedge Blackmails(Jane, John) \qquad \text{[exercise?]}$$

Abstract individuals and reification

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

- involves creating new individuals and new functions for properties/roles
- 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**)

$Purchase(p23) \land agent(p23) = john \land object(p23) = bike \land source(p23) = sears \land amount(p23) = \$200 \land \ldots$

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.*

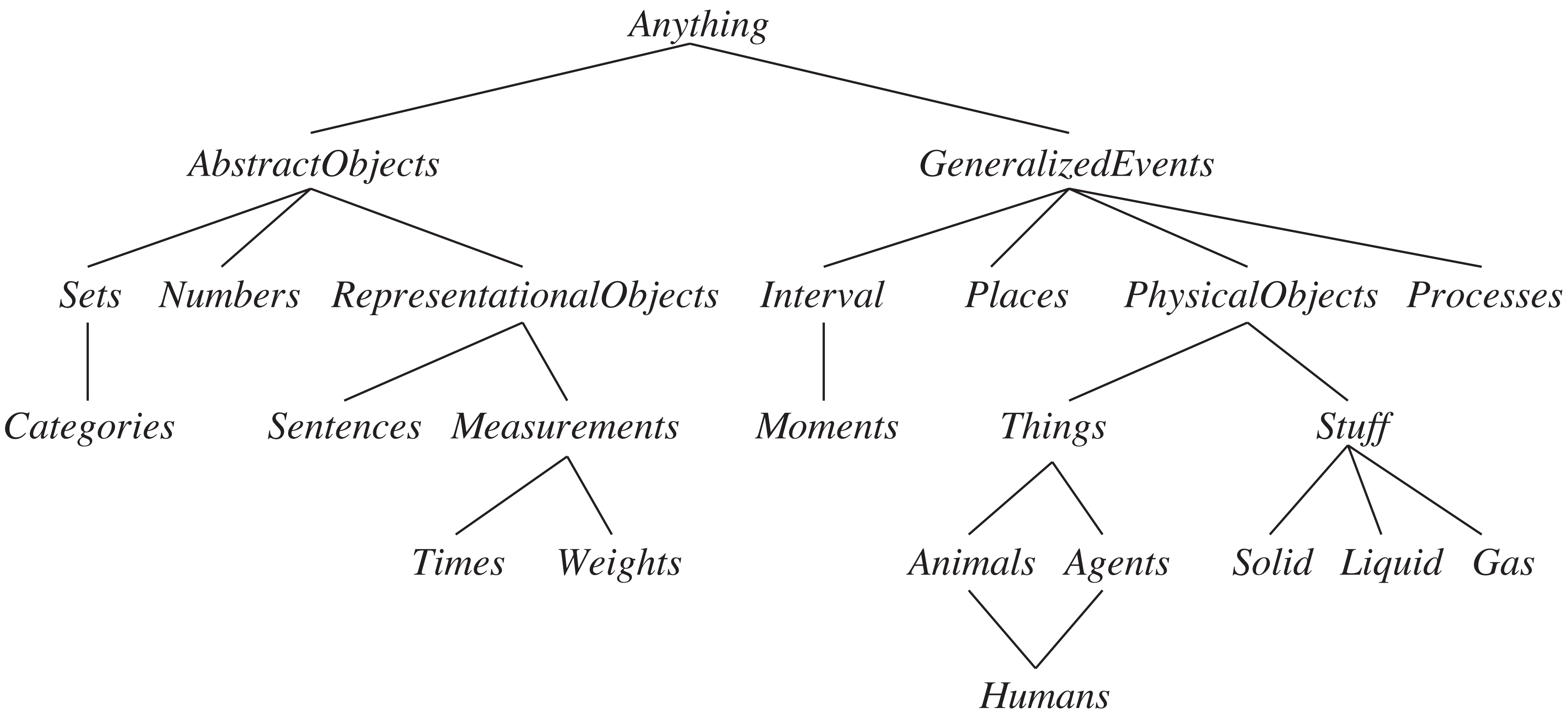
Others ...

Ontology engineering

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 purpose/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.**”
- An upper ontology is like an object-oriented programming framework (reuse)
- Several attempts:
  - CYC (Lenat and Guha, 1990); OpenMind (MIT project); DBpedia (Bizer et al., 2007)
  - 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:

   $WinterSport(Ski)$ $\qquad\qquad\qquad$ $\forall x\ WinterSport(x) \Rightarrow Sport(x)$

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

   $Ski \in WinterSports$

   $WinterSports \subseteq Sports$

This way we can organize categories in **taxonomies** (like in natural sciences), define disjoint categories, partitions … and use specialized inference mechanisms. such as **inheritance**. Description logics take this approach (later).

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:

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

$$PartOf(x, x)$$

Composite objects: structural relations

Structural relations among parts.

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

$$Biped(a) \Rightarrow \exists\ l_1, l_2, b$$
$$Leg(l_1) \land Leg(l2) \land Body(b) \land$$
$$PartOf(l_1, \mathrm{a}) \land PartOf(l_2, \mathrm{a}) \land PartOf(b, a) \land$$
$$Attached(l_1, b) \land Attached(l_2, b) \land$$
$$l_1 \neq l_2 \land [\forall l_3\ Leg(l_3) \land PartOf(l_3, a) \Rightarrow (l_3 = l_1 \lor l_3 = l_2)]$$

exactly two legs!

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. Unlike sets, bunches have weight

$BunchOf(Apples)$ 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 (**logical minimization**).

   $\forall y\, [\forall x\;\; x \in s \Rightarrow PartOf(x, y)] \Rightarrow PartOf(BunchOf(s), y)$

   $BunchOf(s)$ is 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 to these properties are called **measures.**

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

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

L_1

$\forall y\ Centimeters(2.54 \times y) = Inches(y)$

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

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

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

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

where *Time* is for example the UNIX time (seconds elapsed from 1/1/1970)

Qualitative measures

An 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*:

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

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

This is enough to perform some sort of **qualitative inference** and is typical of the field of **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:

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

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**.

Representation and reasoning about states and actions, temporal reasoning

SITUATION CALCULUS, EVENT CALCULUS

The situation calculus in FOL

The situation calculus is a specific ontology in FOL 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 and relations.
- **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

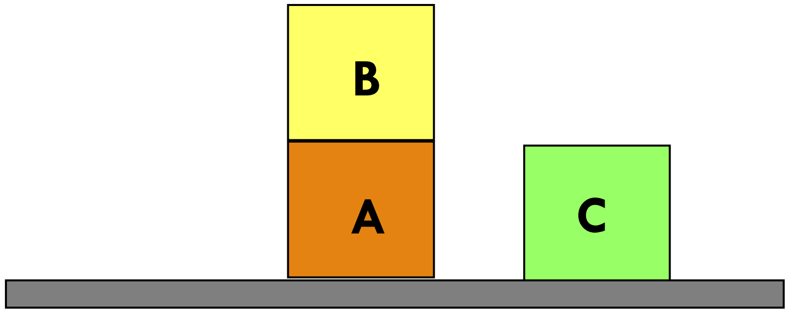
A scenario much used in planning. *The are blocks on a table and the goal is to reach a given arrangement of the blocks by stacking them on top of each other.*

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

  $On, Table, Clear \ldots Hat$ are fluents

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

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

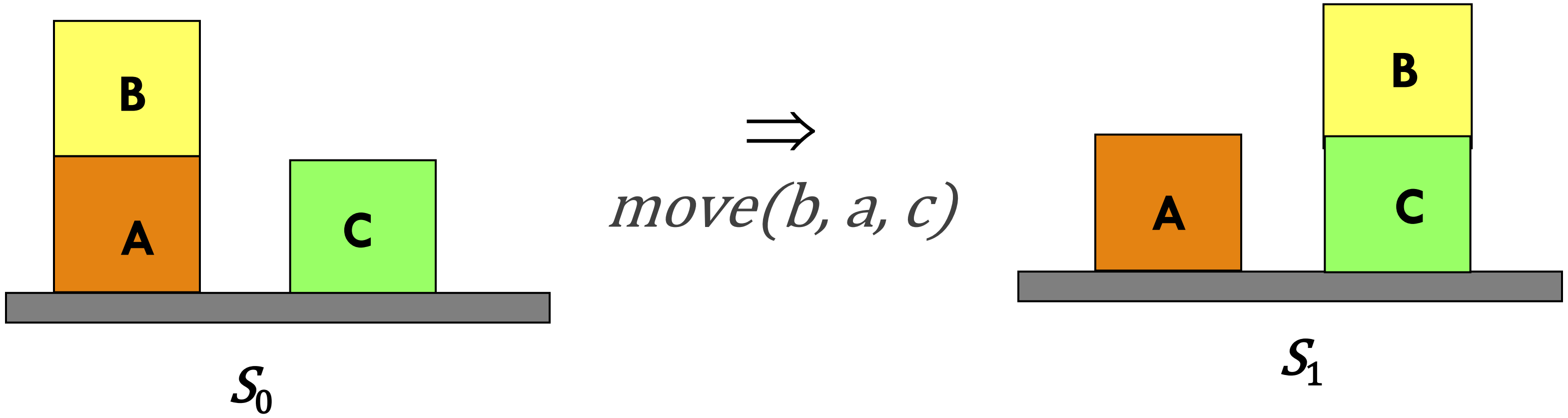
  Immutable properties are represented as before (e.g. $Block$)
- **Actions**: are modelled as functions (terms)

  $move(a, b, c)$

  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. $Result([a\,|\,seq], s) = Result(seq, Result(a, s))$

For example:

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

In general:

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

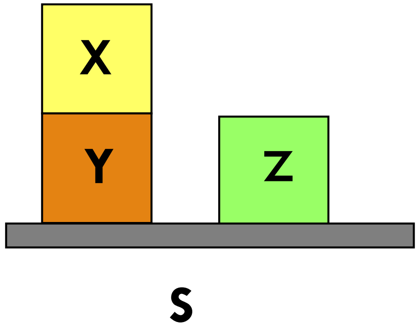
Formalizing actions

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

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

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

  Note: Variables are universally quantified.



- And **effect axioms** such as:

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

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

- This is a specification of the **direct effects** of the action, what changes.
- 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]. The name comes from 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 try to fix the problem writing **frame axioms**.

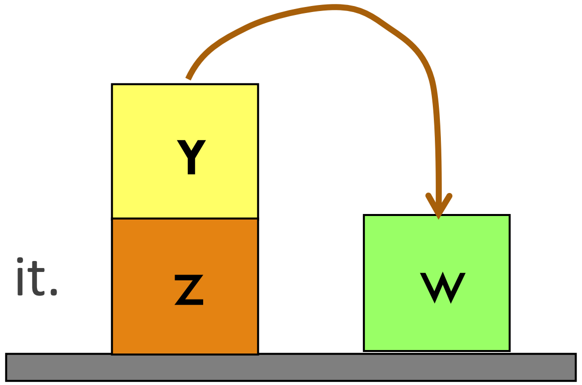
Frame axioms for *Clear* with respect to *move*:

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

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 before* and *an action made f true*] or [*f was true before and no action made it false*]

preconditions
effect
frame axioms

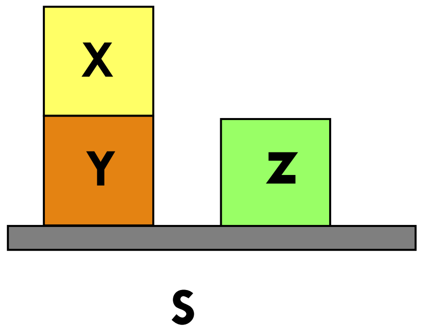
Example: *state-successor* axiom for fluent *Clear*:

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

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

effect $[On(x, y, s) \wedge Clear(x, s) \wedge (a = unstack(x, y))] \vee$

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



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 old location persist for objects.

Uses of situation calculus

Planning: finding a sequence of actions to reach a certain goal condition G

$$KB \vDash \exists \boldsymbol{a}\, G(\mathrm{Result}(\boldsymbol{a}, s_0)) \quad \text{where } \boldsymbol{a} = [a_1, \ldots, a_n]$$

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.

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

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.

Non-monotonic 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 things remain as they were*".

Unfortunately, this leads out of classical logic because it violates the *monotonicity property* 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 non-monotonic reasoning.

In planning we end up using specialized 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 an alternative formalism/ontology known as **event calculus**, was introduced.

Event calculus is based on **events**, **points in time**, **intervals** rather than situations.

Event calculus (in brief)

Event calculus: *reification* of fluents and events

1. A fluent is an object (represented by a function).

   To assert that a **fluent is true** at some point in time t we use the predicate T (True)

   $T(At(Shankar, Berkeley), t)$ $At(Shankar, Berkeley)$ is a term, t a time

   $T(At(Shankar, Berkeley), i)$ $i = (t_1, t_2)$ is a time interval

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

   The event E_1 of Shankar **flying** from San Francisco to Washington is described as

   $E_1 \in Flyings \wedge Flyer(E_1, Shankar) \wedge Origin(E_1, SF) \wedge Destination(E_1, W)$

   To assert that an event happens during an extended period of time we say:

   $Happens(e, i)$

Event calculus ontology

SKIPPED

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

$T(f, t)$	Fluent f is true at time t, or interval
$Happens(e, i)$	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

For intervals:

$Time(x)$	points in a time scale, giving absolute times in seconds (e.g. Unix)
$Begin(i)$, $End(i)$	the earliest and latest moments in an interval
$Duration(i)$	the duration of an interval

Event calculus: some axioms

SKIPPED

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:

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

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:

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

where *Clipped* and *Restored* are properly defined in terms of *Happens, Initiates and Terminates*

Property of intervals:

3. $Interval\,(i) \Rightarrow [Duration(i) = (Time(End(i)) - Time(Begin(i)))]$

Actions in the event calculus

SKIPPED

Actions are modeled as events.

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:

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

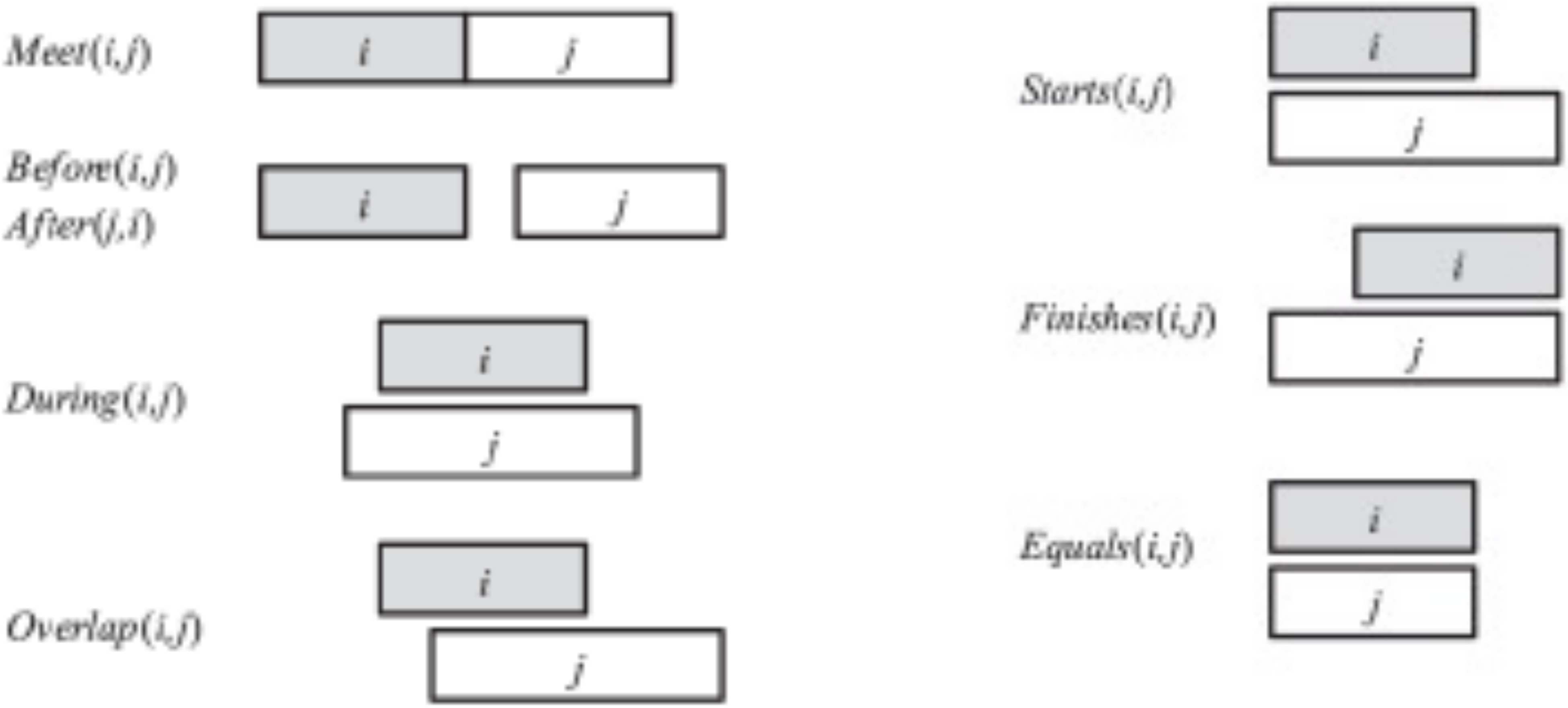
$$Terminates(e, HaveArrow(a), t) \Leftrightarrow e \in Shootings(a)$$

$Start$ is and 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, processes and so on …

Interval relations [Allen 1983]

SKIPPED



Time interval relations

SKIPPED

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

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

Examples:

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

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

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

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

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.
- Formalizing **mental states,** also will lead us to consider non-standard logics.

References

[KRR] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2004 (ch. 3)

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