Agents with Knowledge

5 AGENTS WITH KNOWLEDGE: Outline

♢ Knowledge agents

♢ Logic

♢ Propositional logic

♢ First-order logic

♢ Situation calculus

♢ Logical Agent

♢ Knowledge

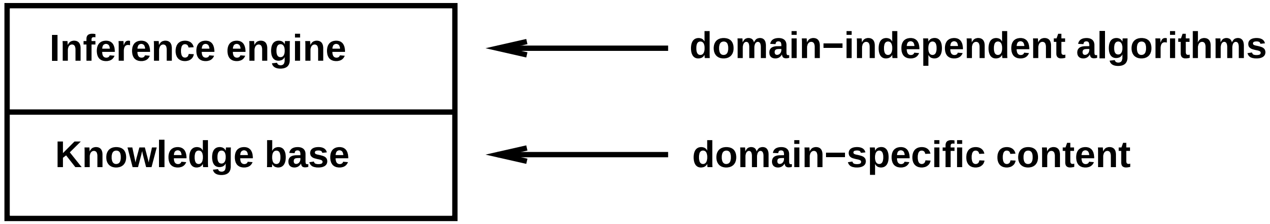
♢ Ontology

♢ Action and change

♢ Mental states

- ♢ Belief
- ♢ Belief-desire-intension
- ♢ Frame, semantic network and inheritance
- ♢ Agents with Commonsense

Knowledge agents



Knowledge base (KB) = set of sentences in a formal language

Declarative approach to building an agent (or other system):
TELL it what it needs to know

Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level
i.e., what they know, regardless of how implemented

Or at the implementation level
i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Logic

Logics are formal languages for representing information
such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the "meaning" of sentences;
i.e., define truth of a sentence in a world

E.g., the language of arithmetic

$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence

$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y

$x + 2 \geq y$ is true in a world where $x = 7,\ y = 1$
$x + 2 \geq y$ is false in a world where $x = 0,\ y = 6$

Types of logic

Logics are characterized by what they commit to as "primitives"

Ontological commitment: what exists—facts? objects? time? beliefs?

Epistemological commitment: what states of knowledge?

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $0 \ldots 1$
Fuzzy logic	degree of truth	degree of belief $0 \ldots 1$

Entailment

Entailment means one thing follows from another

$$KB \models \alpha$$

Knowledge base KB <u>entails</u> sentence α
if and only if
α is true in all worlds where KB is true

E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

Entailment is a relationship between sentences (i.e., *syntax*) that is based on *semantics*

Models

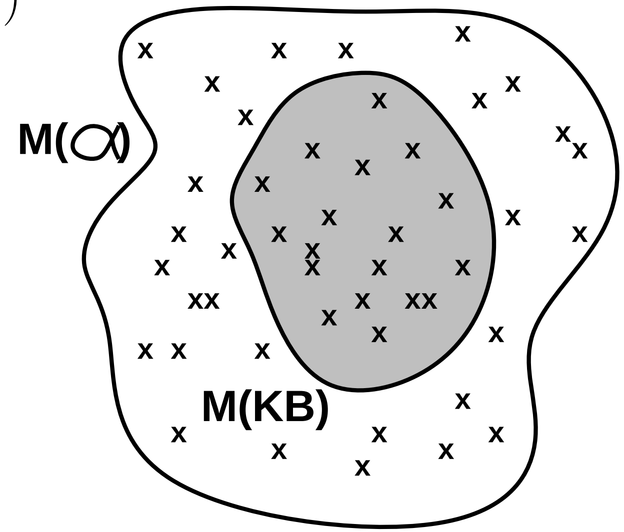
Logicians typically think in terms of <u>models</u>, which are formally structured worlds with respect to which truth can be evaluated

We say m is a <u>model</u> of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

E.g. $KB =$ Giants won and Reds won
$\alpha =$ Giants won



Inference

$KB \vdash_i \alpha =$ sentence α can be derived from KB by procedure i

Soundness: i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Propositional logic: Syntax

Propositional logic (PL) is the simplest logic—illustrates basic ideas

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \wedge S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \vee S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \Rightarrow S_2$ is a sentence

If S_1 and S_2 is a sentence, $S_1 \Leftrightarrow S_2$ is a sentence

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	A	B	C
	$True$	$True$	$False$

Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true <u>and</u>	S_2	is true
$S_1 \vee S_2$	is true iff	S_1	is true <u>or</u>	S_2	is true
$S_1 \Rightarrow S_2$	is true iff	S_1	is false <u>or</u>	S_2	is true
i.e.,	is false iff	S_1	is true <u>and</u>	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true <u>and</u>	$S_2 \Rightarrow S_1$	is true

Propositional inference: Enumeration method

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that $KB \models \alpha$?
Check all possible models—α must be true wherever KB is true

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
$False$	$False$	$False$				
$False$	$False$	$True$				
$False$	$True$	$False$				
$False$	$True$	$True$				
$True$	$False$	$False$				
$True$	$False$	$True$				
$True$	$True$	$False$				
$True$	$True$	$True$				

Truth tables for connectives??

Propositional inference: Solution

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
$False$	$False$	$False$	$False$	$True$	$False$	$False$
$False$	$False$	$True$	$True$	$False$	$False$	$False$
$False$	$True$	$False$	$False$	$True$	$False$	$True$
$False$	$True$	$True$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$True$	$True$	$True$
$True$	$False$	$True$	$True$	$False$	$False$	$True$
$True$	$True$	$False$	$True$	$True$	$True$	$True$
$True$	$True$	$True$	$True$	$True$	$True$	$True$

Inference by enumeration

Truth table enumeration algorithm??

Depth-first enumeration of all models is sound and complete

$O(2^n)$ for n symbols, problem is co-NP-complete

Equivalence

Two sentences are *logically equivalent* iff true in the same models:

$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

12 usual equivalent rules for the connectives

$\alpha \wedge \beta \equiv \beta \wedge \alpha$ (commutativity of $\wedge$) etc.

Validity and Satisfiability

A sentence is <u>valid</u> if it is true in <u>all</u> models

e.g., $A \vee \neg A$, $\quad A \Rightarrow A$, $\quad (A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the <u>Deduction Theorem</u>:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is <u>satisfiable</u> if it is true in <u>some</u> model

e.g., $A \vee B$, $\quad C$

A sentence is <u>unsatisfiable</u> if it is true in <u>no</u> models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable

i.e., prove α by *reductio ad absurdum*

Theorem proving

Proof methods divided (roughly) into two kinds:

Application of inference rules

1. Generation of new sentences from old
2. $Proof$=a sequence of inference rule application
   Can use inference rules as operators in a standard search algorithm
3. Typically require translation of sentences into $normal\ form$

Model checking

1. Truth tables enumerations (always exponential in n)
2. Improved backtracking, e.g., Putnam-Davis
3. Heuristic search in model space (sound but incomplete)
   e.g., the GSAT algorithm

Why FOL: pros and cons of PL

PL is *declarative*: pieces of syntax correspond to facts

PL allows partial/disjuctive/negated informations
(unlike most data structures and databases)

PL is *compositional*:
meaning of $B_{12} \wedge P_{21}$ is derived from meaning of B_{12} and P_{21}

Meaning in PL is *context independent*:
(unlike natural language, where meaning depends on context)

But, PL has very limited expressive power
(unlike natural language)
E.g., cannot say "pits cause breeze in the adjacent squares"
excepts one sentence for each square

First order logic

Whereas propositional logic assumes world contains facts,
first-order logic (like natural language) makes world conceptualization by

1. objects
2. relations (predicate)
3. functions

Syntax of FOL

Let L be a first-order language

	Constants	$KingJohn,\ 2,\ UCB, \ldots$
	Predicates	$Brother,\ >, \ldots$
	Functions	$Sqrt,\ LeftLegOf, \ldots$
Vocabulary:	Variables	$x,\ y,\ a,\ b, \ldots$
	Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
	Equality	$=$
	Quantifiers	$\forall\ \exists$

Atomic sentences

$$\begin{aligned}\text{Atomic sentence} \;&=\; predicate(term_1, \ldots, term_n) \\ &\quad\text{or } term_1 = term_2 \\[1em] \text{Term} \;&=\; function(term_1, \ldots, term_n) \\ &\quad\text{or } constant \text{ or } variable\end{aligned}$$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg{>}(1,2)$

Universal quantification

$\forall \langle variables \rangle \ \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall\, x \ \ At(x, Berkeley) \ \Rightarrow \ Smart(x)$

$\forall\, x \ \ P$ is equivalent to the conjunction of instantiations of P

$$
\begin{array}{ll}
 & At(KingJohn, Berkeley) \ \Rightarrow \ Smart(KingJohn) \\
\wedge & At(Richard, Berkeley) \ \Rightarrow \ Smart(Richard) \\
\wedge & At(Berkeley, Berkeley) \ \Rightarrow \ Smart(Berkeley) \\
\wedge & \ldots
\end{array}
$$

Typically, $\Rightarrow$ is the main connective with $\forall$.
Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall\, x \ \ At(x, Berkeley) \wedge Smart(x)$$

means "Everyone is at Berkeley and everyone is smart"

Existential quantification

$\exists \langle variables \rangle \;\; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists\, x \;\; At(x, Stanford) \land Smart(x)$

$\exists\, x \;\; P$ is equivalent to the disjunction of instantiations of P

$$\begin{array}{ll} & At(KingJohn, Stanford) \land Smart(KingJohn) \\ \lor & At(Richard, Stanford) \land Smart(Richard) \\ \lor & At(Stanford, Stanford) \land Smart(Stanford) \\ \lor & \ldots \end{array}$$

Typically, $\land$ is the main connective with $\exists$.
Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists\, x \;\; At(x, Stanford) \;\Rightarrow\; Smart(x)$$

is true if there is anyone who is not at Stanford!

Properties of quantifiers

$\forall x\ \ \forall y$ is the same as $\forall y\ \ \forall x$ (why??)

$\exists x\ \ \exists y$ is the same as $\exists y\ \ \exists x$ (why??)

$\exists x\ \ \forall y$ is not the same as $\forall y\ \ \exists x$

$\exists x\ \ \forall y\ \ Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y\ \ \exists x\ \ Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x\ \ Likes(x, IceCream) \qquad \neg\exists x\ \ \neg Likes(x, IceCream)$

$\exists x\ \ Likes(x, Broccoli) \qquad \neg\forall x\ \ \neg Likes(x, Broccoli)$

Semantics of FOL

Sentences are true with respect to a <u>model</u> and an <u>interpretation</u>

Model contains objects and relations among them

Interpretation specifies referents for

$constant\ symbols$ → <u>objects</u>

$predicate\ symbols$ → <u>relations</u>

$function\ symbols$ → <u>functional relations</u>

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the <u>objects</u> referred to by $term_1, \ldots, term_n$
are in the <u>relation</u> referred to by $predicate$

Models for FOL

Interpretation I:

the domain $|I|$

1. If σ is an object constant,
   then $\sigma^I \in |I|$
2. If π is an n-ary function constant,
   then $\pi^I : |I|^n \longrightarrow |I|$
3. If ρ is an n-ary relation constant,
   then $\rho^I \subseteq |I|^n$

Models for FOL

Variable assignment U:

a function from the variables of L to objects of $|I|$

Term assignment T_{IU}:

given I and U

1. If τ is an object constant,
   then $T_{IU}(\tau) = I(\tau)$
2. If τ is a variable,
   then $T_{IU}(\tau) = U(\tau)$
3. If τ is a term of the form $\pi(\tau_1, \cdots, \tau)$ and $I(\pi) = g$ and $T_{IU}(\tau_i) = x_i$,
   then $T_{IU}(\tau) = g(x_1, \cdots, x_n)$

Models for FOL

<u>Satisfaction</u> $\models_I \phi[U]$ (simply $\models$):
a sentence ϕ is satisfied by an interpretation I and a variable assignment U

1. $\models (\sigma = \tau)$ iff $T_{IU}(\sigma) = T_{IU}(\tau)$
2. $\models \rho(\tau_1, \cdots, \tau_n)$ iff $< T_{IU}(\tau_1), \cdots, T_{IU}(\tau_n) > \in I(\rho)$
3. $\models \neg\phi$ iff $\not\models \phi$
4. $\models \phi \wedge \psi$ iff $\models \phi$ and $\models \psi$
5. $\models \phi \vee$
   psi iff $\models \phi$ or $\models \psi$
6. $\models \phi \rightarrow \psi$ iff $\not\models \phi$ or $\models \psi$
7. $\models \forall x \phi(x)$ iff for all $d \in |I|$ it is the case that $\models \phi[V]$, where $V(x) = d$ and $V(y) = U(y)$ for $x \neq y$
8. $\models \exists x \phi(x)$ iff for some $d \in |I|$ it is the case that $\models \phi[V]$, where $V(x) = d$ and $V(y) = U(y)$ for $x \neq y$

Models for FOL

Model I:

If an interpretation I satisfies a sentence ϕ for all variable assignments, then I is said to be a $model$ of ϕ, written $\models_I \phi$ or $I \models \phi$

Similarly (in PL), a sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $\quad A \Rightarrow A$, $\quad (A \wedge (A \Rightarrow B)) \Rightarrow B$

A sentence is unsatisfiable if it is true in no models

e.g., $A \wedge \neg A$

Entailment $\models$:

Let Σ be a set of sentences and ϕ a sentence,

$\Sigma \models \phi$ iff ϕ is true in all models of Σ

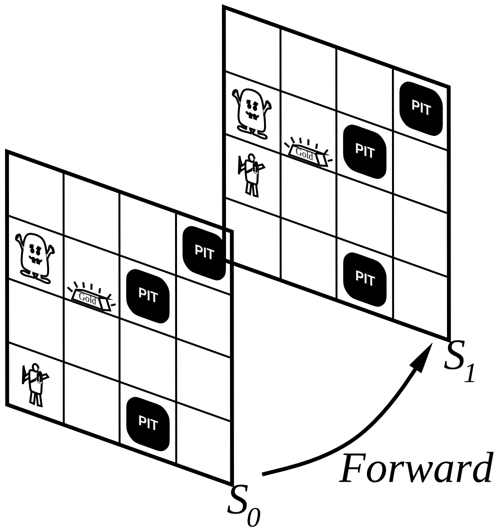
Situation Calculus

Facts hold in <u>situations</u>, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

<u>Situation calculus</u> is one way to represent change in FOL:

Adds a situation argument to each non-eternal predicate
E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a is s



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

Frame problem: find an elegant way to handle non-change

(a) representation—avoid frame axioms
(b) inference—avoid repeated “copy-overs” to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Actions

Successor-state axioms solve the representational frame problem

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

$$\begin{array}{lcl} \text{P true afterwards} & \Leftrightarrow & [\text{an action made P true} \\ & \vee & \text{P true already and no action made P false}] \end{array}$$

For holding the gold:

$$\begin{aligned} &\forall\, a, s \;\; Holding(Gold, Result(a,s)) \;\Leftrightarrow \\ &\quad [(a = Grab \wedge AtGold(s)) \\ &\quad \vee (Holding(Gold, s) \wedge a \neq Release)] \end{aligned}$$

Making plans

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $\textsc{Ask}(KB, \exists\, s \;\; Holding(Gold, s))$
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 $\textsc{Ask}(KB, \exists\, p \;\; Holding(Gold, PlanResult(p, S_0)))$
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

Planning in situation calculus

$PlanResult(p, s)$ is the situation resulting from executing p in s

$$PlanResult([], s) = s$$
$$PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Initial state $At(Home, S_0) \land \neg Have(Milk, S_0) \land \ldots$

Actions as Successor State axioms

$Have(Milk, Result(a, s)) \;\Leftrightarrow$
$[(a = Buy(Milk) \land At(Supermarket, s)) \lor (Have(Milk, s) \land a \neq \ldots)]$

Query

$$s = PlanResult(p, S_0) \land At(Home, s) \land Have(Milk, s) \land \ldots$$

Solution

$$p = [Go(Supermarket), Buy(Milk), Buy(Bananas), Go(HWS), \ldots]$$

Principal difficulty: unconstrained branching, hard to apply heuristics

Logic Agent

Wumpus agent
- The wumpus world Knowledge Base
- Finding pits and wumpus using logical inference
- Translating knowledge into action

Circuit-based agent

situation calculus based agent

Knowledge

Knowledge:

-*Language*, e.g., FOL
-*Representation*, e.g., declarative knowledge
-*Reasoning*, e.g., proofs and model checking

The separation between the knowledge base and reasoning procedure should be maintained

Knowledge base (KB): a good KB should be expressive, concise, unambiguous, context-insensitive, effective, clear and correct

Knowledge engineering (expert systems, knowledge-based systems): the process of building a knowledge base
The knowledge engineer or agent usually interview the real experts or environments to become educated about the domain and to elicit required knowledge in a process called knowledge acquisition

Knowledge engineering vs. programming

	Knowledge engineering	Programming
1.	Choosing a logic	Choosing a programming language
2.	Building a knowledge base	Writing a program
3.	Implementing the proof theory	Choosing or writing a compiler
4.	Inferring new facts	Running a program

Should be less work

Ontology

<u>Ontology</u>: a vocabulary for the domain knowledge

<u>Ontological engineering</u>: representing various ontology

The five-step methodology

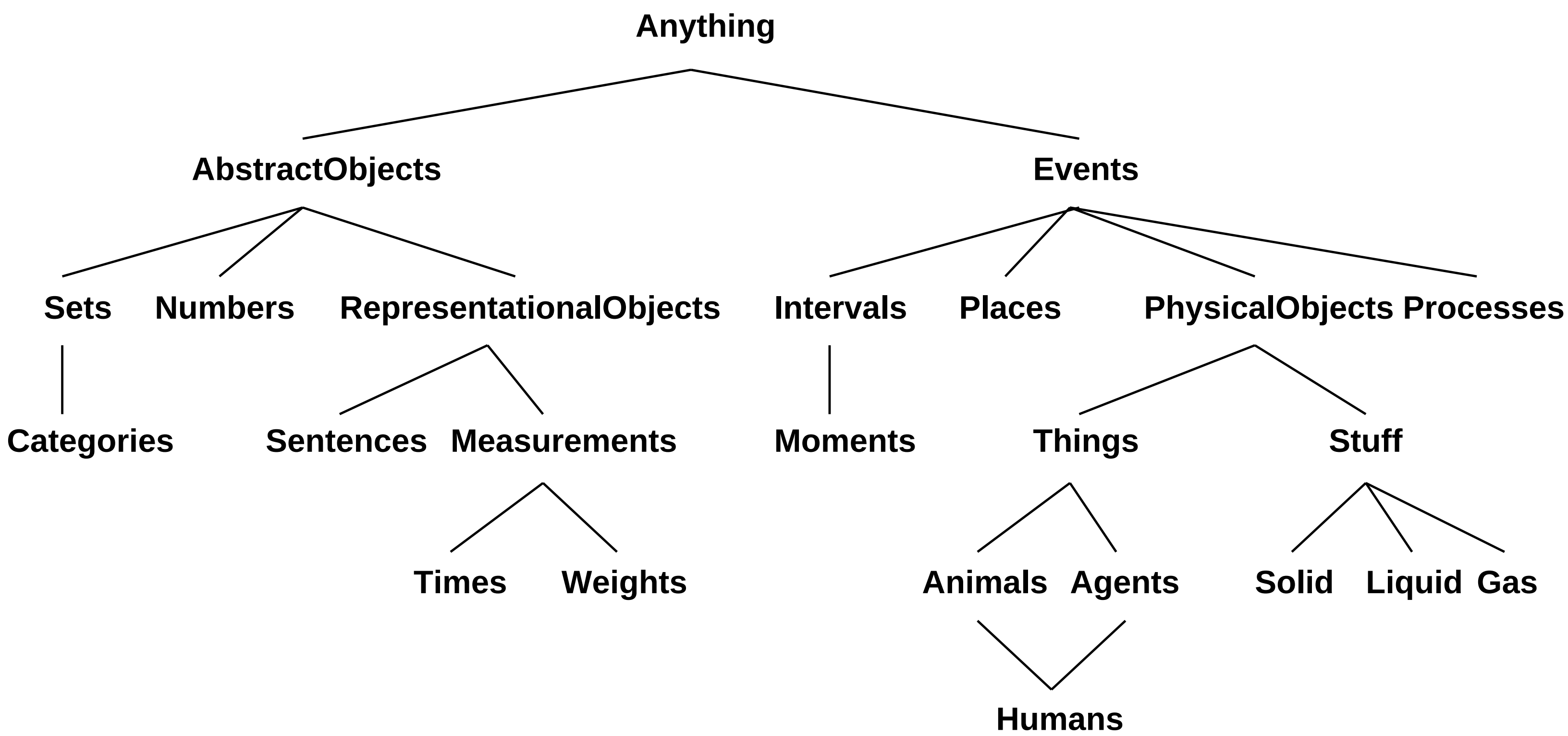
1. Decide what to talk about
2. Decide on a vocabulary of predicates, functions and constants
3. Encode general knowledge about the domain
4. Encode a description of the specific problem instance
5. Pose queries to the inference procedure and get answers

General ontology

A general-purpose ontology has advantages over special-purpose one

- ♢ Categories
- ♢ Measures
- ♢ Composite objects
- ♢ Time, Space, and Change
- ♢ Events and Processes
- ♢ Physical objects
- ♢ Substances
- ♢ Mental objects and belief

The world ontology



Categories

Category: include as members all objects having certain properties
E.g., An object (penguin) is a member of a category (birds)
$Penguin \in Birds$

Subclass relations organize categories into a taxonomy (hierachy)
E.g., a category is a subclass of another category
$Tomatoes \in Fruit$

Inheritance: the individual inherits the property of the category from their membership
E.g., $Child(x, y) \wedge Familyname(John, y) \rightarrow Familyname(John, x)$

The problem: natural kind or inheritance with exception
E.g., $\forall x. x \in Typical(Bird) \Rightarrow Flies(x)$

Description logic for categories

Description logic: focus on categories and their definitions

- *Subsumption*: checking if one category is a subset of another based on their definitions
- *Classification*: checking if an object belongs to a category

Action and change

Time:

E.g., $At(Evening, Sleep)$

Event:

E.g., $WorldWarII, SubEvent(BattleOfBritain, WorldWarII)$

An event that includes as subevents all events occuring in a given time period is called interval

Space:

E.g., $In(Beijing, China)$

$\forall xl. Location(x) = l \Leftrightarrow$
$At(x, l) \wedge \forall l_1 At(x, l_1) \Rightarrow In(l, l_1)$

Process: liquid event

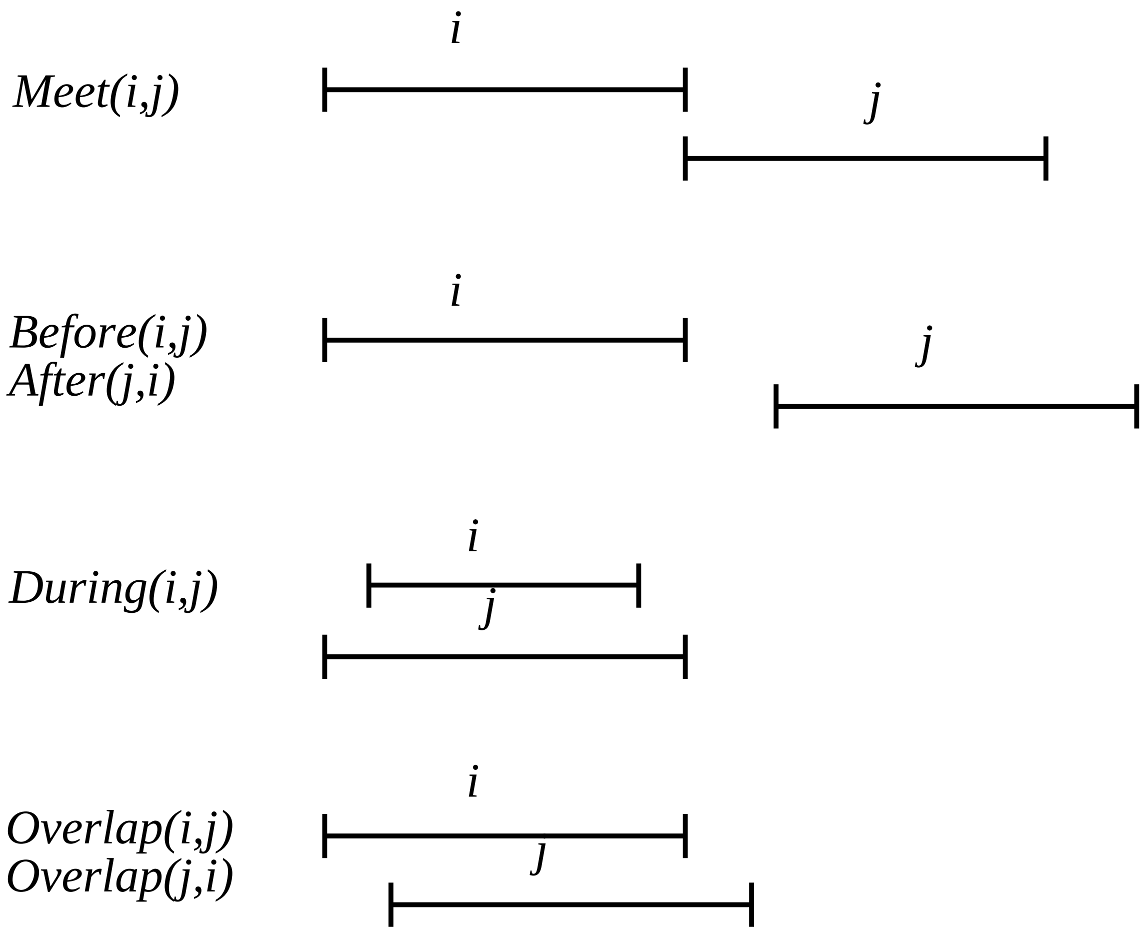
E.g., $T(Working(Teacher), TodayLessonHours)$

$T(c, i)$ means that some event of type c occured over exactly the interval i

Action and change contd.

Time interval:

E.g., $\forall ij.Meet(i,j) \Leftrightarrow Time(End(i)) = Time(Start(j))$



Action and change contd.

Action:

E.g., $\forall xyi_0.T(Engaged(x,y),i_0) \Rightarrow$
$\exists i_1(Meet(i_0,i_1) \vee After(i_1,i_0)) \wedge$
$T(Marry(x,y) \vee BreakEngagement(x,y),i_1)$

Fluent: something that changes across situations

E.g., $President(USA)$
$T(Democrat(President(USA)),AD2003)$

Context:

E.g., $President(USA,AD2003) = GeorgeWBush$

Mental states

Propositional attitudes (modalities): e.g., know, believe, want, expect, etc.

Multi-agents: e.g., an agent reasons about the mental processes of the other agents

Formalizing reasoning about mental states:
- -syntactic theory
- -possible worlds (modal logic)

Modal operators: B, K

$B(a, \psi)$ or $B_a(\psi)$: agent a believes that sentence ψ is true

$K(a, \psi)$ or $K_a(\psi)$: agent a knows that sentence ψ is true

$B(A, \psi), A = \{a_1, \cdots, a_n\}$: every agent of A believes that sentence ψ is true

Belief: A formal theory

Extending first-order language L:

$Belief$ formulas: $Believes(Agent, fluent)$

$Strings$: $Flies(Clark)$ represented as $[F, l, i, e, s, (, C, l, a, r, k,),]$
-referential opaque: an equal term cannot be substituted for the one (mental object) in the scope of belief, e.g., $"Clark" \neq "Superman"$

Den function: mapping a string to the object that it denotes

$Name$ function: mapping an object to a string that is the name of a constant that denotes the object

E.g.,

$Den("Clark") = ManOfSteel \land Den("Superman") = ManOfSteel$

$Name(ManOfSteel) = K_{11}$

Belief contd.

Inference rules,e.g., Modus Ponens

$\forall apq. LogicalAgent(a) \wedge Believes(a,p) \wedge Believes(a, Concat(p, " \Rightarrow ", q) \Rightarrow Believes(a,q)$

where $Concat$ is a function on strings that concatenates their elements together, abbreviate $Concat(p, " \Rightarrow ", q)$ as "$\underline{\mathsf{p}}\Rightarrow\underline{\mathsf{q}}$"

E.g., belief rules, -if a logical agent believes something, then it believes that it believes it

$\forall ap. LogicalAgent(a) \wedge Believes(a,p) \Rightarrow Believes(a, "Believes(\underline{\mathsf{Name(a)}}, \underline{\mathsf{p}})")$

Belief contd.

Logical omniscience:

$Believes(a, \phi), Believes(a, \phi \Rightarrow \psi) \models Believes(a, \psi)$

-So we need limited rational agent

Belief and knowledge: knowledge is justified true belief

$\forall ap.Knows(a, p) \Leftrightarrow Believes(a, p) \wedge T(Den(p) \wedge T(Den(KB(a)) \Rightarrow Den(p))$

Belief and Time: $Believes(agent, string, interval)$

Knowledge and action: knowledge producing actions

Belief-desire-intension

The *Belief-Desire-Intention* (BDI) model of agent targets to discloses the internal structure of an intelligent agent further. It explains the process of agent's decision-making.

Belief: agent's mental reflection of outside world and its physical state.

Desire: the goals the agent desire to achieve.

Intention: the actions that the agent intends to perform to satisfy its desires.

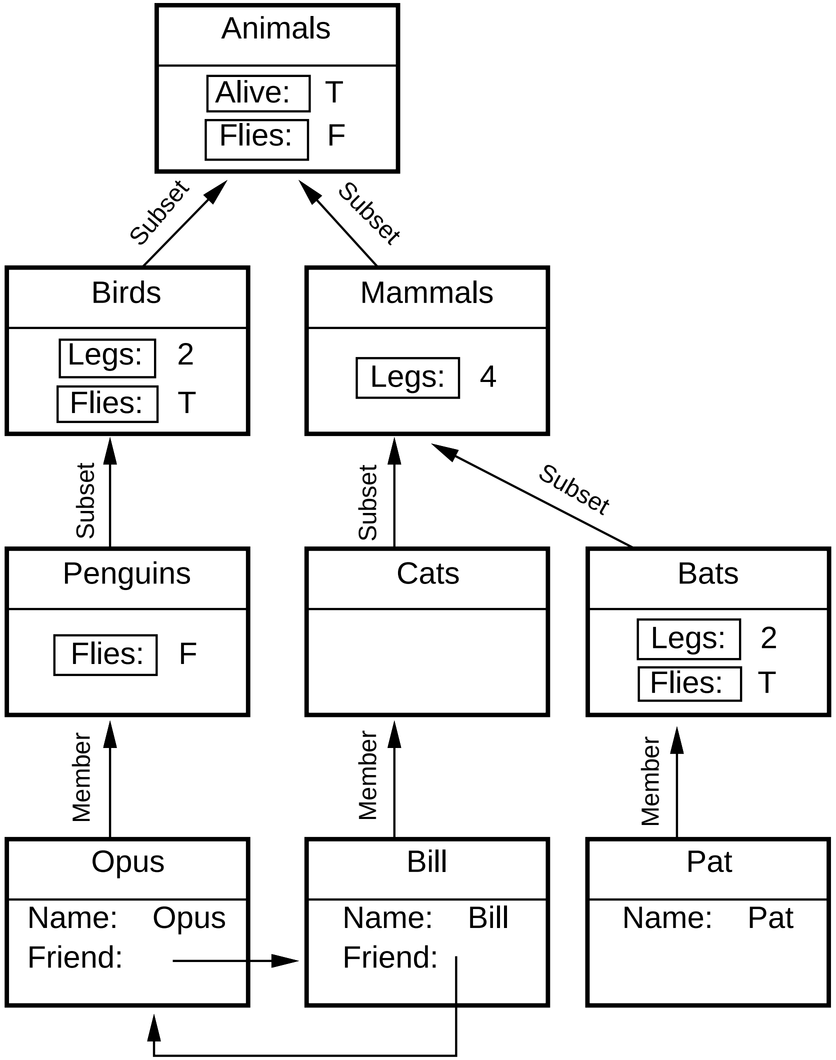
Example:

I believe that if I work hard I will pass this course.

I desire to pass this course.

I intend to work hard.

Frame, semantic network and inheritance



(a) A frame−based knowledge base

Rel(Alive,Animals,T)
Rel(Flies,Animals,F)

Birds $\subset$ Animals
Mammals $\subset$ Animals

Rel(Flies,Birds,T)
Rel(Legs,Birds,2)
Rel(Legs,Mammals,4)

Penguins $\subset$ Birds
Cats $\subset$ Mammals
Bats $\subset$ Mammals

Rel(Flies,Penguins,F)
Rel(Legs,Bats,2)
Rel(Flies,Bats,T)

Opus $\in$ Penguins
Bill $\in$ Cats
Pat $\in$ Bats

Name(Opus,"Opus")
Name(Bill,"Bill")
Friend(Opus,Bill)
Friend(Bill,Opus)
Name(Pat,"Pat")

(b) Translation into first−order logic

Frame contd.

Link Type	Semantics	Example
$A \xrightarrow{Subset} B$	$A \subset B$	$Cats \subset Mammals$
$A \xrightarrow{Member} B$	$A \in B$	$Bill \in Cats$
$A \xrightarrow{R} B$	$R(A, B)$	$Bill \xrightarrow{Age} 12$
$A \xrightarrow{\boxed{R}} B$	$\forall x \;\; x \in A \;\Rightarrow\; R(x, B)$	$Birds \xrightarrow{\boxed{Legs}} 2$
$A \xrightarrow{\boxed{\boxed{R}}} B$	$\forall x \;\; \exists y \;\; x \in A \;\Rightarrow\; y \in B \wedge R(x, y)$	$Birds \xrightarrow{\boxed{\boxed{Parent}}} Birds$

Inheritance

Inheritance with exceptions

$$\forall rxb.Holds(r,x,b) \Leftrightarrow$$
$$Val(r,x,b) \vee (\exists px \in p \wedge Rel(r,p,b) \wedge \neg InterveningRel(x,p,r))$$

$$\forall xpr.InterveningRel(x,p,r) \Leftrightarrow$$
$$\exists i Intervening(x,i,p) \wedge \exists b' Rel(r,i,b')$$

$$\forall aip.Intervening(x,i,p) \Leftrightarrow (x \in i) \wedge (i \subset p)$$

Multiple inheritance

Commonsense reasoning

The example

KB:

$\forall x Bird(x) \Rightarrow Flies(x)$
$Bird(Tweety)$

$\underline{\underline{KB \vdash Flies(Tweety)}}$??

With exceptions:

$\forall x Bird(x) \wedge x \neq Penguin \wedge \cdots \Rightarrow Flies(x)$

$\forall x Bird(x) \wedge \neg Abnormal(x) \Rightarrow Flies(x)$

Commonsense contd.

The problem

Monotonicity of FOL:

if $KB \vdash P$ then $(KB \wedge S) \vdash P$
i.e., if P follows from KB, then it still follows when KB is augmented by $TELL(KB, S)$

Nonmonotonicity: $KB \subset KB', \exists P, KB \vdash P$ but $KB' \not\vdash P$
Nonmonotonic logic is the formalization of reasoning with incomplete knowledge

Agent with incomplete knowledge

Closed World Assumption (CWA)

Let KB be a (finite) set of sentence (belief set), $T(KB)$ theory of KB ($T(KB) = \{\phi | KB \models \phi\}$)

The CWA of KB, written as $CWA(KB) = KB \cup KB_{asm}$, defined as follows:

1. $\phi \in T(KB)$ iff $KB \models \phi, \phi$ is a sentence
2. $\neg p \in KB_{asm}$ iff $p \notin T(KB), p$ is a ground atom
3. $\phi \in CWA(KB)$ iff $\{KB \cup KB_{asm}\} \models \phi$

Agent with incomplete knowledge contd.

CWA

$KB = \{p(A), p(A) \Rightarrow q(A), p(B)\}$

$T(KB) \not\models q(B), T(KB) \not\models \neg q(B)$

$CWA(KB) \models \neg q(B)$

The problem

$KB = \{p(A) \vee p(B)\}$

$CWA(KB) \models \neg p(A) \wedge \neg p(B)$

Web shopping agent