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 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;
\[ x^2 + y > \] 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”


Epistemological commitment: what states of knowledge?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0...1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0...1</td>
</tr>
</tbody>
</table>
Entailment means one thing follows from another

\[ KB \models \alpha \]

Knowledge base $KB$ entails sentence $\alpha$

if and only if

$\alpha$ 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
Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$.

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

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

E.g. $KB = \text{Giants won and Reds won}$

$\alpha = \text{Giants won}$
Inference

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

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, $S_1 \land S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \lor S_2$ 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. \[ \begin{array}{ccc}
A & B & C \\
True & True & False
\end{array} \]

Rules for evaluating truth with respect to a model \( m \):

\[
\begin{array}{ll}
\neg S & \text{is true iff } S \text{ is false} \\
S_1 \land S_2 & \text{is true iff } S_1 \text{ is true and } S_2 \text{ is true} \\
S_1 \lor S_2 & \text{is true iff } S_1 \text{ is true or } S_2 \text{ is true} \\
S_1 \Rightarrow S_2 & \text{is true iff } S_1 \text{ is false or } S_2 \text{ is true} \\
i.e., & \text{is false iff } S_1 \text{ is true and } S_2 \text{ is false} \\
S_1 \Leftrightarrow S_2 & \text{is true iff } S_1 \Rightarrow S_2 \text{ is true and } S_2 \Rightarrow S_1 \text{ is true}
\end{array}
\]
Propositional inference: Enumeration method

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

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

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$A \lor C$</th>
<th>$B \lor \neg C$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Truth tables for connectives??
# Propositional inference: Solution

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$A \lor C$</td>
<td>$B \lor \lnot C$</td>
<td>$KB$</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
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
Two sentences are *logically equivalent* iff true in the same models:
\[ \alpha \equiv \beta \iff \alpha \models \beta \quad \text{and} \quad \beta \models \alpha \]

12 usual equivalent rules for the connectives

\[ \alpha \land \beta \equiv \beta \land \alpha \quad \text{(commutativity of } \land \text{) etc.} \]
Validity and Satisfiability

A sentence is **valid** if it is true in all models
\[ A \lor \neg A, \quad A \Rightarrow A, \quad (A \land (A \Rightarrow B)) \Rightarrow B \]

Validity is connected to inference via the **Deduction Theorem**:
\[ KB \models \alpha \text{ if and only if } (KB \Rightarrow \alpha) \text{ is valid} \]

A sentence is **satisfiable** if it is true in some model
\[ A \lor B, \quad C \]

A sentence is **unsatisfiable** if it is true in no models
\[ A \land \neg A \]

Satisfiability is connected to inference via the following:
\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]
i.e., prove \( \alpha \) 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/disjunctive/negated informations
(Unlike most data structures and databases)

PL is **compositional**:
meaning of $B_{12} \land P_{21}$ 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

**Vocabulary:**
- **Constants**: $KingJohn$, 2, $UCB$, ...
- **Predicates**: $Brother$, $>$, ...
- **Functions**: $Sqrt$, $LeftLegOf$, ...
- **Variables**: $x$, $y$, $a$, $b$, ...
- **Connectives**: $\wedge$, $\lor$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
- **Equality**: $=$
- **Quantifiers**: $\forall$, $\exists$
Atomic sentences

Atomic sentence  =  \textit{predicate}(\textit{term}_1, \ldots, \textit{term}_n)

or \textit{term}_1 = \textit{term}_2

\textit{Term}  =  \textit{function}(\textit{term}_1, \ldots, \textit{term}_n)

or \textit{constant} or \textit{variable}

E.g.,  \textit{Brother}((\textit{KingJohn}, \textit{RichardTheLionheart})

> (\textit{Length}((\textit{LeftLegOf} (\textit{Richard})), \textit{Length}((\textit{LeftLegOf} (\textit{KingJohn}))))}
Complex sentences

Complex sentences are made from atomic sentences using connectives

$$
\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2
$$

E.g. $\text{Sibling}(\text{KingJohn, Richard}) \Rightarrow \text{Sibling}(\text{Richard, KingJohn})$

$$
>(1, 2) \lor \leq (1, 2) \\
>(1, 2) \land \neg>(1, 2)
$$
Universal quantification

\[ \forall \langle \text{variables} \rangle \langle \text{sentence} \rangle \]

Everyone at Berkeley is smart:
\[ \forall x \text{ At}(x, \text{Berkeley}) \Rightarrow \text{Smart}(x) \]

\[ \forall x \ P \] is equivalent to the conjunction of instantiations of \( P \)

\[ \text{At}(\text{KingJohn}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{KingJohn}) \]
\[ \land \text{At}(\text{Richard}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{Richard}) \]
\[ \land \text{At}(\text{Berkeley}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{Berkeley}) \]
\[ \land \ldots \]

Typically, \( \Rightarrow \) is the main connective with \( \forall \).
Common mistake: using \( \land \) as the main connective with \( \forall \):

\[ \forall x \ \text{At}(x, \text{Berkeley}) \land \text{Smart}(x) \]

means “Everyone is at Berkeley and everyone is smart”
Existential quantification

\[ \exists \langle \text{variables} \rangle \langle \text{sentence} \rangle \]

Someone at Stanford is smart:
\[ \exists x \ At(x, \text{Stanford}) \land \text{Smart}(x) \]

\[ \exists x \ P \] is equivalent to the disjunction of instantiations of \( P \)

\[ \begin{align*}
& At(\text{KingJohn}, \text{Stanford}) \land \text{Smart}(\text{KingJohn}) \\
\lor & At(\text{Richard}, \text{Stanford}) \land \text{Smart}(\text{Richard}) \\
\lor & At(\text{Stanford}, \text{Stanford}) \land \text{Smart}(\text{Stanford}) \\
\lor & \ldots
\end{align*} \]

Typically, \( \land \) is the main connective with \( \exists \).

Common mistake: using \( \Rightarrow \) as the main connective with \( \exists \):
\[ \exists x \ At(x, \text{Stanford}) \Rightarrow \text{Smart}(x) \]

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

\( \forall x \ Likes(x, \text{IceCream}) \) \( \implies \neg \exists x \ \neg Likes(x, \text{IceCream}) \)

\( \exists x \ Likes(x, \text{Broccoli}) \) \( \implies \neg \forall x \ \neg Likes(x, \text{Broccoli}) \)
Semantics of FOL

Sentences are true with respect to a model and an interpretation.

Model contains objects and relations among them.

Interpretation specifies referents for:
- constant symbols → objects
- predicate symbols → relations
- function symbols → functional relations

An atomic sentence \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \) is true iff the objects referred to by \( \text{term}_1, \ldots, \text{term}_n \) are in the relation referred to by \( \text{predicate} \).
Models for FOL

Interpretation $I$:

the domain $|I|$

1. If $\sigma$ is an object constant,
   then $\sigma^I \in |I|$
2. If $\pi$ is an $n$-ary function constant,
   then $\pi^I : |I|^n \rightarrow |I|$
3. If $\rho$ 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 $\tau$ is an object constant,
   then $T_{IU}(\tau) = I(\tau)$

2. If $\tau$ is a variable,
   then $T_{IU}(\tau) = U(\tau)$

3. If $\tau$ 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

**Satisfaction** $\models_I \phi[U]$ (simply $\models$):

- a sentence $\phi$ is satisfied by an interpretation $I$ and a variable assignment $U$

1. $\models (\sigma = \tau)$ iff $T_{IU}(\sigma) = T_{IU}(\tau)$
2. $\models \rho(\tau_1, \ldots, \tau_n)$ iff $< T_{IU}(\tau_1), \ldots, T_{IU}(\tau_n) > \in I(\rho)$
3. $\models \neg \phi$ iff $\not\models \phi$
4. $\models \phi \land \psi$ iff $\models \phi$ and $\models \psi$
5. $\models \phi \lor \psi$ iff $\models \phi$ or $\models \psi$
6. $\models \phi \rightarrow \psi$ iff $\models \phi$ 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 $\phi$ for all variable assignments, then $I$ is said to be a model of $\phi$, written $|= I \phi$ or $I \models \phi$.

Similarly (in PL), a sentence is valid if it is true in all models:
e.g., $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

A sentence is unsatisfiable if it is true in no models:
e.g., $A \land \neg A$

Entailment $|=:$
Let $\Sigma$ be a set of sentences and $\phi$ a sentence,
$\Sigma \models \phi$ iff $\phi$ is true in all models of $\Sigma$.
Situation Calculus

Facts hold in situations, rather than eternally
E.g., Holding(Gold, Now) rather than just Holding(Gold)

Situation calculus 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, . . .
Successor-state axioms solve the representational frame problem

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

\[ P \text{ true afterwards} \iff [\text{an action made } P \text{ true} \lor P \text{ true already and no action made } P \text{ false}] \]

For holding the gold:

\[ \forall a, s \; Holding(Gold, Result(a, s)) \iff [(a = Grab \land AtGold(s)) \lor (\text{Holding}(Gold, s) \land a \neq Release)] \]
Initial condition in KB:
\[ \text{At}(\text{Agent}, [1, 1], S_0) \]
\[ \text{At}(\text{Gold}, [1, 2], S_0) \]

Query: \text{Ask}(KB, \exists s \text{ Holding(Gold, s)})

i.e., in what situation will I be holding the gold?

Answer: \{s/\text{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.
Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\)

\(\text{PlanResult}(p, s)\) is the result of executing \(p\) in \(s\)

Then the query \(\text{Ask}(KB, \exists p \ \text{Holding}(\text{Gold, PlanResult}(p, S_0)))\)
has the solution \(\{p/\text{[Forward, Grab]}\}\)

Definition of \(\text{PlanResult}\) in terms of \(\text{Result}\):
\[
\forall s \ \text{PlanResult}([\ ], s) = s \\
\forall a, p, s \ \text{PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))
\]

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 \(s\)

\[
\begin{align*}
PlanResult([], s) &= s \\
PlanResult([a\mid p], s) &= PlanResult(p, Result(a, s))
\end{align*}
\]

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

**Actions as Successor State axioms**

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

**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

<table>
<thead>
<tr>
<th>Knowledge engineering</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choosing a logic</td>
<td>Choosing a programming language</td>
</tr>
<tr>
<td>2. Building a knowledge base</td>
<td>Writing a program</td>
</tr>
<tr>
<td>3. Implementing the proof theory</td>
<td>Choosing or writing a compiler</td>
</tr>
<tr>
<td>4. Inferring new facts</td>
<td>Running a program</td>
</tr>
</tbody>
</table>

Should be less work
Ontology: a vocabulary for the domain knowledge

Ontological engineering: 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
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
Categories

**Category**: include as members all objects having certain properties

E.g., An object (penguin) is a member of a category (birds)

\[ \text{Penguin} \in \text{Birds} \]

Subclass relations organize categories into a taxonomy (hierarchy)

E.g., a category is a subclass of another category

\[ \text{Tomatoes} \in \text{Fruit} \]

**Inheritance**: the individual inherits the property of the category from their membership

E.g., \( \text{Child}(x, y) \land \text{Familyname}(\text{John}, y) \rightarrow \text{Familyname}(\text{John}, x) \)

The problem: natural kind or inheritance with exception

E.g., \( \forall x. x \in \text{Typical}(\text{Bird}) \Rightarrow \text{Flies}(x) \)
**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 occurring in a given time period is called **interval**

Space:
E.g., \(In(Beijing, China)\)
\[\forall xl. Location(x) = l \iff At(x, l) \land \forall l_1 At(x, l_1) \Rightarrow In(l, l_1)\]

Process: liquid event
E.g., \(T(Working(Teacher), TodayLessonHours)\)
\(T(c, i)\) means that some event of type \(c\) occurred over exactly the interval \(i\)
Action and change contd.

Time interval:
E.g., $\forall i,j. \text{Meet}(i,j) \Leftrightarrow \text{Time}(\text{End}(i)) = \text{Time}(\text{Start}(j))$

<table>
<thead>
<tr>
<th>Meet$(i,j)$</th>
<th>$i$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before$(i,j)$</td>
<td>$i$</td>
<td>$j$</td>
</tr>
<tr>
<td>After$(j,i)$</td>
<td>$i$</td>
<td>$j$</td>
</tr>
<tr>
<td>During$(i,j)$</td>
<td>$i$</td>
<td>$j$</td>
</tr>
<tr>
<td>Overlap$(i,j)$</td>
<td>$i$</td>
<td>$j$</td>
</tr>
<tr>
<td>Overlap$(j,i)$</td>
<td>$i$</td>
<td>$j$</td>
</tr>
</tbody>
</table>
Action and change contd.

Action:
E.g., \( \forall xyi_0. T(\text{Engaged}(x,y), i_0) \Rightarrow \exists i_1(\text{Meet}(i_0, i_1) \lor \text{After}(i_1, i_0)) \land \)
\( T(\text{Marry}(x,y) \lor \text{BreakEngagement}(x,y), i_1) \)

Fluent: something that changes across situations
E.g., \( \text{President}(\text{USA}) \)
\( T(\text{Democrat}(\text{President}(\text{USA})), \text{AD}2003) \)

Context:
E.g., \( \text{President}(\text{USA}, \text{AD}2003) = \text{GeorgeWBush} \)
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

\[ B(a, \psi) \text{ or } B_a(\psi): \text{agent } a \text{ believes that sentence } \psi \text{ is true} \]

\[ K(a, \psi) \text{ or } K_a(\psi): \text{agent } a \text{ knows that sentence } \psi \text{ is true} \]

\[ B(A, \psi), A = \{a_1, \ldots, a_n\}: \text{every agent of } A \text{ believes that sentence } \psi \text{ is true} \]
Belief: A formal theory

Extending first-order language $L$:

**Belief formulas:** $\text{Believes}(\text{Agent}, \text{fluent})$

**Strings:** $\text{Flies}(\text{Clark})$ 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.,

$\text{Den} (”\text{Clark}”) = \text{ManOfSteel} \land \text{Den} (”\text{Superman}”) = \text{ManOfSteel}$

$\text{Name}(\text{ManOfSteel}) = K_{11}$
Inference rules, e.g., Modus Ponens

\[ \forall a p q. \text{LogicalAgent}(a) \land \text{Believes}(a, p) \land \text{Believes}(a, \text{Concat}(p, " \Rightarrow ", q)) \Rightarrow \text{Believes}(a, q) \]

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

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

\[ \forall a p. \text{LogicalAgent}(a) \land \text{Believes}(a, p) \Rightarrow \text{Believes}(a, " \text{Believes}(\text{Name}(a), p)" ) \]
Belief contd.

**Logical omniscience:**

\[ \text{Believes}(a, \phi), \text{Believes}(a, \phi \Rightarrow \psi) \models \text{Believes}(a, \psi) \]

-So we need limited rational agent

**Belief and knowledge:** knowledge is justified true belief

\[ \forall a p. \text{Knows}(a, p) \iff \text{Believes}(a, p) \land T(\text{Den}(p) \land T(\text{Den}(KB(a)) \Rightarrow \text{Den}(p))) \]

**Belief and Time:** \text{Believes}(agent, string, interval)

**Knowledge and action:** knowledge producing actions
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 ⊂ Animals
Mammals ⊂ Animals

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

Penguins ⊂ Birds
Cats ⊂ Mammals
Bats ⊂ Mammals

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

Opus ∈ Penguins
Bill ∈ Cats
Pat ∈ Bats

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

(b) Translation into first-order logic
### Frame contd.

<table>
<thead>
<tr>
<th>Link Type</th>
<th>Semantics</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \subset B$</td>
<td>$A \subseteq B$</td>
<td>$\text{Cats} \subseteq \text{Mammals}$</td>
</tr>
<tr>
<td>$A \in B$</td>
<td>$A \in B$</td>
<td>$\text{Bill} \in \text{Cats}$</td>
</tr>
<tr>
<td>$A \xrightarrow{R} B$</td>
<td>$R(A, B)$</td>
<td>$\text{Bill} \xrightarrow{\text{Age}} 12$</td>
</tr>
<tr>
<td>$\forall x \ x \in A \Rightarrow R(x, B)$</td>
<td>$\forall x \ x \in A \Rightarrow R(x, B)$</td>
<td>$\text{Birds} \xrightarrow{\text{Legs}} 2$</td>
</tr>
<tr>
<td>$\forall x \ \exists y \ x \in A \Rightarrow y \in B \land R(x, y)$</td>
<td>$\forall x \ \exists y \ x \in A \Rightarrow y \in B \land R(x, y)$</td>
<td>$\text{Birds} \xrightarrow{\text{Parent}} \text{Birds}$</td>
</tr>
</tbody>
</table>
Inheritance

Inheritance with exceptions

∀rxb.\textit{Holds}(r, x, b) \iff
Val(r, x, b) \lor (\exists px \in p \land \textit{Rel}(r, p, b) \land \neg\textit{InterveningRel}(x, p, r))

∀xr.\textit{InterveningRel}(x, p, r) \iff
\exists i\textit{Intervening}(x, i, p) \land \exists b' \textit{Rel}(r, i, b')

∀ai.\textit{Intervening}(x, i, p) \iff (x \in i) \land (i \subset p)

Multiple inheritance
Commonsense reasoning

The example

KB:

\[ \forall x \text{Bird}(x) \Rightarrow \text{Flies}(x) \]
\[ \text{Bird}(\text{Tweety}) \]

\[ KB \vdash \text{Flies}(\text{Tweety}) ?? \]

With exceptions:

\[ \forall x \text{Bird}(x) \land x \neq \text{Penguin} \land \cdots \Rightarrow \text{Flies}(x) \]
\[ \forall x \text{Bird}(x) \land \neg \text{Abnormal}(x) \Rightarrow \text{Flies}(x) \]
The problem

Monotonicity of FOL:

if $KB \vdash P$ then $(KB \land S) \vdash P$

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
Closed World Assumption (CWA)

Let $KB$ be a (finite) set of sentence (belief set), $T(KB)$ theory of $KB$

\[ T(KB) = \{ \phi \mid KB \models \phi \} \]

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

\[ KB = \{p(A) \vee p(B)\} \]
\[ CWA(KB) \models \neg p(A) \land \neg p(B) \]
Web shopping agent