Description Logics and Frames

- Considerable interest in this topics
  - seen as starting point for representing huge quantities of knowledge
  - often used for representing terminologies in particular domains
  - any increase of level of automation will give rise to an increased significance in domain description

- Semantic web:
  OWL DL (Web Ontology Language/Description Logic): formal semantics (model theory) and support for reasoning

http://www.w3.org/TR/owl2-overview/
Basic ideas are old ...
Ontologies

- **Ontology**: science which investigates and explains the nature and essential properties and relations of all beings (Aristotle: Metaphysica)

- In AI, artifact that
  - makes use of a vocabulary,
  - with a set of rules about syntax and meaning,
  - purpose: computer-interpretable description of a domain

- Typical building blocks:
  - **names** (concepts)
  - **relations and constraints**

- Popular applications: in biomedical research (e.g. http://bioontology.org/)
Linnaeus

Ordo 1.

PRIMATES.

Dentes primores superiores IV parallel.
Mammæ pectorales, binae.

1. HOMO nosce Te ipsum.

1. H. diurnus. (*), vagans cultura, loco.
   a. H. rufus, cholericus, rectus.
   b. H. albus, fangoineus torofus.
   c. H. luridus, melancholicus rigidus.
   d. H. niger, phlegmaticus, laxus.
   e. H. monstrolos folo (a.), vel arte (b. c.)
      a. Alpini parvi, agiles, timidi: Patagonii magni, feaces.
      b. Monarchides ut minus ferriles: Hottentotti.
      d. Macerepulis capite conico: Chinenses.
      e. Plagipalpis capite antice compresso: Canadenses.

2. Homo nocturnus. Ourang Outang. Bont. faw. 84. t. 84.

Genus Troglodite seu Ourang Outang ab Homine vero diffici-
sum, adhibita quamvis omni attentione, obtinere non potui, nisi affer-
merem notam lubricam, in aliis generibus non constat. Nec Dextrae
lanarii minus a reliquis remoti, nec Nympheæ eaffræ, quibus
carent Simiae, bene ad Simias redurre admittiebant. Inquiram en-
texta in viciss, qua ratione, modo noto aliquae exstans, ab Homine
gerere separari quocum, nam inter Simias veritatem oportet esse Simi-
liam. Apollodor.
Cycorp is a leading provider of semantic technologies that bring a new level of intelligence and common sense reasoning to a wide variety of software applications. The Cyc® software combines an unparalleled common sense ontology and knowledge base with a powerful reasoning engine and natural language interfaces to enable the development of novel knowledge-intensive applications.

As a premier knowledge-based technologies research and development company, Cycorp leverages its cutting edge innovations in knowledge representation, machine reasoning, natural language processing, semantic data integration, and information management and search to offer an array of semantic middleware, knowledge-based application development capabilities, and turn-key solutions.

The Cyc project, a long-term quest to develop a true artificial intelligence, was founded in 1984 by Dr. Douglas Lenat as a lead project in the Microelectronics and Computer Technology Corporation (MCC). In 1994, Cycorp was founded to further develop, commercialize, and apply the Cyc technology. To foster the growth of semantic reasoning by the research community, Cycorp offers a no-cost license to its semantic technologies development toolkit to the research community. Additionally, it has placed the core Cyc ontology into the public domain.
Knowledge server

Start time: Wed Sep 30 23:27:42 CEST 2009
Lisp implementation: Cycclp Java SubL Runtime Environment
JVM: Sun Microsystems Inc. Java HotSpot(TM) Server VM 1.6.0_03 (1.6.0_03-b05)
Current KB: 5018
Patch Level: 10.128401
Running on: knuth
OS: Linux 2.6.23.17-88.fc7 (i386)
Working directory: /home/peterl/Research/Cyc/opencyc-2.0/server/cyc/run
Total memory allocated to VM: 1348MB.
Memory currently used: 541MB.
Memory currently available: 806MB.
Initializing HL backing store caches from units/5018/.
;; At this point the cyc http server is running and you can access
;; Cyc directly via the local web browser.
;; http://localhost:3602/cgi-bin/cycgci/cg?cb-start
;; You can browse cyc via the Guest account or perform updates by
;; logging on as CycAdminstrator.
CYC(1):
Representation of relations

1. **Semantic nets:**
   - syntactic: nodes and relations between nodes
   - represented as a graph
   - semantics of nodes and relationships

2. **Frames:**
   - (binary) relations represented as attributes
   - inheritance
   - subtyping

3. **Description logic:**
   - concepts
   - binary relationships
   - restrictions
Basics of description logics

Example DL:

\[ ALC = \text{Attribute concept Language with Complement} \]

Basic ingredients:

- concepts
- roles
- Boolean operators

“A man is married to a doctor, and all of whose children are either doctors or professors”

\[
\text{Human} \sqcap \neg \text{Female} \sqcap \exists \text{married.Doctor}) \sqcap (\forall \text{hasChild.}(\text{Doctor} \sqcup \text{Professor}))
\]
Language elements

Concept descriptions:
- primitive concept $C$, e.g., Human, $\top$ (most general), $\bot$ (empty)
- primitive role $r$, e.g., hasChild
- conjunction $\sqcap$, e.g., SmartHuman $\sqcap$ Student
- disjunction $\sqcup$, e.g., Truck $\sqcup$ Car
- complement $\neg$, e.g., $\neg$Human
- value restriction $\forall r. C$, e.g., $\forall$hasChild.Doctor
- existential restriction $\exists r. C$, e.g., $\exists$happyChild.Parent

All understood in terms of (groups of) individuals and properties of individuals
General concept inclusion (GCI) also called subsumption

For concepts $C, D$:

- $C \sqsubseteq D$, e.g., Professor $\sqsubseteq$ Person

- **definition** $C \equiv D$ is the same as $C \sqsubseteq D$ and $D \sqsubseteq C$ (Not $(C \sqsubseteq D) \sqcap (D \sqsubseteq C)$, why?)

  - $C$ in $C \equiv D$ is called a **defined** concept, whereas $D$ consists of primitive concepts, e.g.,

    - Father $\equiv \neg \text{Female} \sqcap \exists \text{hasChild}.\text{Human}$
Concrete descriptions

Instances of concepts or roles, called assertions:

- \( c : C \), means that \( c \) is an instance of concept \( C \), e.g.,
  
  John : Person

- \( (b, d) : r \), means that the pair of individuals \( (b, d) \) is an instance of role \( r \), e.g.,
  
  (John, Mary) : marriedTo
Knowledge base

Knowledge Base = KB

Terminology = TBox
Father ≡ ¬Female ⊓ ∃hasChild.Human
Human ⊑ Animal

Concrete assertions = ABox
John : Father
(John, Sheila) : hasChild

KB = (TBox, ABox)
Knowledge base

KB = (TBox, ABox):

- TBox: contains general descriptions, definitions, subsumptions relationships (GCIs)
  Father ≡ ¬Female ⊓ ∃hasChild.Human
  Human ⊑ Animal

- ABox: contains description of individuals (instances)
  John : Father
  (John, Sheila) : hasChild

- Sometimes ABox = ∅, then only interested in general principles: terminological reasoning
Meaning of description logic

In terms of set theory

Let $\mathcal{I} = (\Delta, \cdot)$ be an interpretation, then

- $\top^\mathcal{I} = \Delta$, and $\bot^\mathcal{I} = \emptyset$
- each concept $C^\mathcal{I} \subseteq \Delta$
- each role $r^\mathcal{I} \subseteq \Delta \times \Delta$
- $(C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
- $(C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
- $(\exists r.C)^\mathcal{I} = \{c \in \Delta \mid \exists d \in C^\mathcal{I} \text{ with } (c, d) \in r^\mathcal{I}\}$
- $(\forall r.C)^\mathcal{I} = \{c \in \Delta \mid \forall d \in \Delta, \text{ if } (c, d) \in r^\mathcal{I} \text{ then } d \in C^\mathcal{I}\}$

where $C^\mathcal{I}$ and $r^\mathcal{I}$ are interpretations of $C$ and $r$ as sets.
Example

Father $\equiv \neg$ Female $\sqcap \exists$ hasChild.Human

Interpretation $\mathcal{I} = (\Delta, \cdot )$, with $\Delta = \{\text{John, Sheila}\}$

- Father$^\mathcal{I} = \{\text{John}\} \subseteq \Delta$
- Human$^\mathcal{I} = \{\text{John, Sheila}\}$
- hasChild$^\mathcal{I} = \{(\text{John, Sheila})\}$
- $(\exists$ hasChild.Human$)^\mathcal{I} = \{\text{John}\}$
Meaning of subsumption

Interpretation $\mathcal{I} = (\Delta, \cdot)$, then

$$(C \subseteq D)^\mathcal{I} = C^\mathcal{I} \subseteq D^\mathcal{I}$$

Example:

Father $\subseteq$ Human

$\Delta = \{\text{John, Sheila}\}$, then

$\text{Father}^\mathcal{I} = \{\text{John}\}$

and

$\text{Human}^\mathcal{I} = \{\text{John, Sheila}\}$
Relationship with predicate logic

Translation function $\tau_x$ (DL $\rightarrow$ first-order predicate logic) that introduces variable $x$:

- $\tau_x(C') = C(x)$
- $\tau_x(C \cap D) = (\tau_x(C') \land \tau_x(D'))$
- $\tau_x(C \cup D) = (\tau_x(C') \lor \tau_x(D'))$
- $\tau_x(\exists r.C') = \exists y \ r(x, y) \land \tau_y(C')$
- $\tau_x(\forall r.C') = \forall y \ r(x, y) \rightarrow \tau_y(C')$

Example: $\neg \text{Female} \cap \exists \text{hasChild}.\text{Human}$:

$\tau_x(\neg \text{Female} \cap \exists \text{hasChild}.\text{Human})$
$= \tau_x(\neg \text{Female}) \land \tau_x(\exists \text{hasChild}.\text{Human})$
$= \neg \text{Female}(x) \land \tau_x(\exists \text{hasChild}.\text{Human})$
$= \neg \text{Female}(x) \land \exists y(\text{hasChild}(x, y) \land \text{Human})(y)$
Relationship with predicate logic

GCIs $C \sqsubseteq D$ in TBox:

$$\bigwedge_{C \subseteq D \in \text{TBox}} \forall x (\tau_x(C) \rightarrow \tau_x(D))$$

Thus, $\sqsubseteq$ becomes logical implication

Example: Translate

$\text{UnivTeacher} \sqsubseteq \text{Prof} \sqcup \neg \text{Undergraduate}$

to predicate logic:

$\tau_x (\text{UnivTeacher} \sqsubseteq \text{Prof} \sqcup \neg \text{Undergraduate})$

$= \forall x (\text{UnivTeacher}(x) \rightarrow \tau_x (\text{Prof} \sqcup \neg \text{Undergraduate}))$

$= \forall x (\text{UnivTeacher}(x) \rightarrow (\text{Prof}(x) \lor \neg \text{Undergraduate}(x)))$
Relationship with predicate logic

Translation of ABox:

\[
\tau_x(\text{ABox}) = \bigwedge_{c:C \in \text{ABox}} \tau_c(C) \land \bigwedge_{(b,d):r \in \text{ABox}} r(b, d)
\]

Thus, unit clauses of the form \( C(c) \) and \( r(b, d) \)

Example:

\[\text{ABox} = \{\text{John} : \text{Father}, (\text{John}, \text{Sheila}) : \text{hasChild}\}\]

In first-order logic:

\[\text{Father}(\text{John}) \land \text{hasChild}(\text{John}, \text{Sheila})\]
Reasoning

Possible procedure:

- Translate DL knowledge base to first-order logic
- Use a reasoning engine, e.g., resolution, for reasoning, then

\[ \tau(KB) \vdash \phi \]

with \( \phi \) something like \( \tau_x(\text{Prof} \sqsubseteq \text{Human}) \) becomes:

\[ \tau(KB) \land \neg\phi \vdash \bot \]

(i.e., if KB is consistent and KB \( \cup \neg\phi \) is inconsistent, then \( \phi \) follows from KB)

- However, special purpose reasoning may be advantageous
Typical questions

Let $\text{KB} = (\text{TBox}, \text{ABox})$, then:

- $\text{KB} \models C \sqsubseteq D$? (is $C$ subsumed by $D$?)
- $\text{KB} \models c : C$? (is $c$ an instance of $C$?)
- $\text{KB} \models (b, d) : r$? (is the pair $(b, d)$ true for role $r$?)

Reasoning can be reduced to consistency checking:

- $(\text{TBox}, \text{ABox} \cup \{c : C \sqcap \neg D\}) \vdash \bot$
- $(\text{TBox}, \text{ABox} \cup \{c : \neg C\}) \vdash \bot$
- $(\text{TBox}, \text{ABox} \cup \{(b, d) : \neg r\}) \vdash \bot$
Summary DL

Description logics: restricted logical languages for the representation of conceptual information and terminologies

Basis for all large-scale attempts for the development of knowledge bases, e.g. semantic web and biomedical ontologies

Advantages:
- Restricted syntax allows developing special purpose, efficient reasoning systems (e.g., Racer, Fact, Cyc)
- \( ALC \) is decidable! (a 2 variable fragment of first-order logic)

Disadvantage: many things cannot be represented in a DL
Frame formalism

Predecessor of description logics, still in use

Concepts and properties of concepts

Example (human anatomy): “An artery is a vessel. An artery transports blood from the heart to the tissues, and is characterised by a thick wall and much muscular tissue.”

Frame taxonomy: hierarchical organisation

- **Nodes** (vertices): classes and instances
- **Arcs** (arrows): superclasses and instance-of relationships
Syntax

Basic elements:

- subclass relationship between frames
- attributes or slots
- values or fillers

⇒ attribute-value pair or slot-filler combination

Example: (structure, tube) is an attribute-value pair

```
class vessel is
  superclass nil;
  structure = tube;
  contains = blood
end
```

```
class artery is
  superclass vessel;
  wall = muscular
end
```

```
instance aorta is
  instance-of artery;
  diameter = 3cm
end
```
Meaning of frames

Semantics in terms of **predicate logic**:

**class** $C$ **is** superclass $S$;

\[
\forall x (C(x) \rightarrow S(x))
\]

\[
\forall x (C(x) \rightarrow a_1(x) = b_1)
\]

\[
\vdots
\]

\[
\forall x (C(x) \rightarrow a_n(x) = b_n)
\]

end

**instance** $i$ **is** instance-of $C$;

\[
C(i)
\]

\[
d_1(i) = e_1
\]

\[
\vdots
\]

\[
d_m(i) = e_m
\]

end
Relationship with description logic

class $C$ is superclass $S$;

\[
a_1 = b_1; \\
\vdots \\
a_n = b_n
\]
end

(instance $i$ is instance-of $C$;

\[
d_1 = e_1; \\
\vdots \\
d_m = e_m
\]
end

TBox:

\[
C \sqsubseteq S
\]

\[
C \sqsubseteq \exists a_1.\{b_1\}
\]

\[\vdots\]

\[
C \sqsubseteq \exists a_n.\{b_n\}
\]

\[
\{i\} \text{ is a concept if } i \text{ is an individual}
\]

ABox:

\[
i : C
\]

\[
(i, e_1) : d_1
\]

\[\vdots\]

\[
(i, e_m) : d_m
\]
Basic reasoning method is called **inheritance**, sometimes example of **non-monotonic** reasoning.

Frame inherits attribute-value pairs from its generalisations.

Two types of inheritance:

1. **single**: tree-shaped taxonomy
2. **multiple**: general graph-shaped taxonomy (loops)
**Single inheritance**

Requirement: unique attribute names

**Example**

```
function Inherit(frame, a-v-pairs)
    if frame = nil
        then return(a-v-pairs)
    fi;
    a-v-pairs ← a-v-pairs ∪ AttributePart(frame);
    return(Inherit(superframe(frame), a-v-pairs))
end
```

\[ F \text{ inherits } a_1 = v_1, a_2 = v_2, a_3 = v_3 \]
Exceptions

Non-unique attribute names ⇒ exceptions / inconsistency

Example
class artery is
  instance-of artery;
  superclass vessel;
  blood = oxygenrich
end

instance pulmonary-a is
  instance-of artery;
  blood = oxygenpoor
end

∀x(artery(x) → vessel(x))
∀x(artery(x) → blood(x) = oxygenrich)
artery(pulmonary-a)
blood(pulmonary-a) = oxygenpoor

Logically inconsistent!

Solution: local ‘overwriting’
Single inheritance with exceptions

Search the taxonomy until a value is found:

```
function Inherit(frame, a-v-pairs)
    if frame = nil then return (a-v-pairs) fi;
    pairs ← AttributePart(frame);
    a-v-pairs ← a-v-pairs ∪ NewAttributes(pairs, a-v-pairs);
    return (Inherit(superframe(frame), a-v-pairs))
end
```

Example

```
F inherits a = c
```

Diagram:

- `F`
- `a = d`
- `a = c`
- `a = e`
- `F` inherits `a = c`
Multiple inheritance with exceptions

Questions:

1. Taxonomy consistent?
2. If yes, what are the inherited values for attribute $a$ vof $y_3$?

Requirement: solution must work for both: graph and tree shaped taxonomies!
Monotonic reasoning

- Inheritance with exceptions example of non-monotonic reasoning

- Monotonic reasoning:
  - knowledge base $\text{KB}$
  - add knowledge to $\text{KB}$ and obtain new knowledge base $\text{KB}'$
  - if $\text{KB} \vdash \text{Results}$ and $\text{KB}' \vdash \text{Results'}$ then $\text{Results} \subseteq \text{Results'}$
  - more knowledge yields more results

- Example:

  $\text{KB} = \{ P \rightarrow Q \}$ \hspace{1cm} $\text{KB}' = \{ P \rightarrow Q, P \}$
  
  $\text{Results} = \text{KB}$ \hspace{1cm} $\text{Results'} = \text{KB}' \cup \{ Q \} = \text{KB} \cup \{ P, Q \}$
Non-monotonic reasoning

Non-monotonic reasoning is more close related to human reasoning

- knowledge base KB
- add knowledge to KB and obtain new knowledge base KB'
- if KB ⊢ Results and KB' ⊢ Results' then Results ⊆ Results' does not hold in general
- more knowledge does not necessarily yield more results

Example (|∼ is non-monotonic):
\[
\{ \text{artery(left-pulmonary-artery)}, \\
\text{blood(left-pulmonary-artery)} = \text{oxygen-poor}, \\
\forall x (\text{artery}(x) \rightarrow \text{blood}(x) = \text{oxygen-rich}) \}
\]
|∼ blood(left-pulmonary-artery) = oxygen-poor