Lecture 16 of 42

Knowledge Engineering and Ontology Engineering
Discussion: Description Logics

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:

Sections 10.1 – 10.2, p. 320 – 327, Russell & Norvig 2nd edition
http://en.wikipedia.org/wiki/Ontology_(information_science)

Lecture Outline

- Reading for Next Class: Sections 10.1 – 10.2 (p. 272 – 319), R&N 2e
- Last Class: Resolution Theorem Proving, 9.5 (p. 275-294), R&N 2e
  - Proof example in detail
  - Paramodulation and demodulation
  - Resolution strategies: unit, linear, input, set of support
  - FOL and computability: complements (different difficulty) and duals (same)
  - Theoretical foundations and ramifications of decidability results
- Today: Prolog in Brief, Knowledge Engineering (KE), Ontologies
  - Prolog examples
  - Introduction to ontologies
    - Description logics and the Web Ontology Language (OWL)
    - Ontologies defined and ontology design
- Next Class: More Ontology Design; Situation Calculus Revisited
  - Knowledge engineering (KE) and knowledge management
  - KR and reasoning about states, actions, properties
- Coming Week: Ontologies, Description Logics, Semantic Nets

Acknowledgements

Professor Ian Horrocks

Professor of Computer Science
Oxford University
Computing Laboratory
Fellow, Oriel College

© 2006

Horrocks, I.
Oxford University
(formerly University of Manchester)
http://en.wikipedia.org/wiki/Ian_Horrocks

Stuart Russell

Professor of Computer Science
Chair, Department of Electrical Engineering and Computer Sciences
Smith-Zadeh Professor in Engineering
Computer Science Division
387 Soda Hall
University of California
Berkeley, CA 94720-1776

© 2004-2005

Russell, S. J.
University of California, Berkeley
http://www.eecs.berkeley.edu/~russell/

Peter Norvig
Director of Research Google

Norvig, P.
http://norvig.com/

Slides from:
http://aima.eecs.berkeley.edu

Logic Programming (Prolog) Systems: Review

Basis: backward chaining with Horn clauses + bells & whistles
Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques ⇒ approaching a billion LIPS

Program = set of clauses = head :- $\texttt{literal}_1$, ... $\texttt{literal}_n$.

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
```

Efficient unification by open coding
Efficient retrieval of matching clauses by direct linking
Depth-first, left-to-right backward chaining
Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
Closed-world assumption ("negation as failure")
e.g., given `alive(X) :- not dead(X).`
`alive(joe)` succeeds if `dead(joe)` fails

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Prolog Examples in Depth: Review

Depth-first search from a start state X:

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

No need to loop over S: `successor` succeeds for each

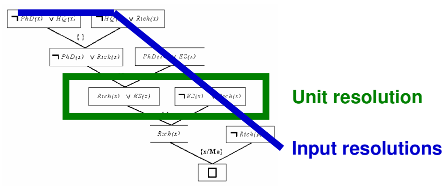
Appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).
```

```
query:    append(A,B,[1,2]) ?
answers: A=[]     B=[1,2]
         A=[1]    B=[2]
         A=[1,2]  B=[]
```

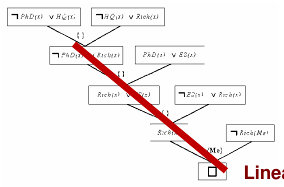

Unit and Input Resolution: Review

- **Unit Preference**
  - **Idea: Prefer inferences that produce shorter sentences**
  - **Compare: Occam's Razor**
  - **How? Prefer unit clause (*single-literal*) resolvents ($\alpha \vee \beta$ with $\neg \beta \vee \alpha$)**
  - **Reason: trying to produce a short sentence ($\perp \equiv$ True $\Rightarrow$ False)**
- **Input Resolution**
  - **Idea: "diagonal" proof (proof "list" instead of proof tree)**
  - **Every resolution combines some input sentence with some other sentence**
  - **Input sentence: *in original KB or query***



Linear Resolution and Set-of-Support: Review

- **Linear Resolution**
  - **Generalization of input resolution**
  - **Include any *ancestor in proof tree* to be used**

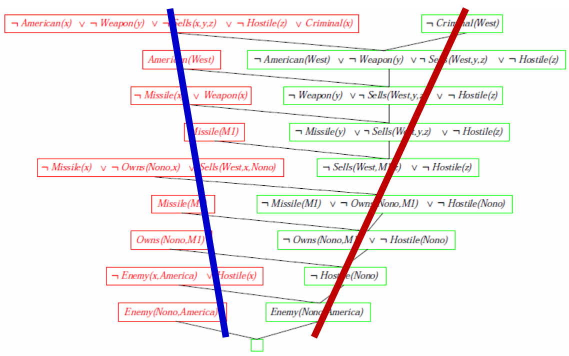


Linear resolutions

- **Set of Support (SoS)**
  - **Idea: try to eliminate some potential resolutions**
  - **Prevention as opposed to cure**
  - **How?**
    - **Maintain set SoS of resolution results**
    - **Always take *one resolvent* from it**
  - **Caveat: need right choice for SoS to ensure completeness**

Subsumption: Review

- **Subsumption**
  - **Idea: eliminate sentences that sentences that are more specific than others**
  - **e.g., *P*(*x*) subsumes *P*(*A*)**
- **Putting It All Together**



Semi-Decidability of L_{VALID} & L_{SAT}^{C}: Review

- $L_{FOL\text{-}VALID}$ (written L_{VALID}): Language of Valid Sentences (Tautologies)
- Deciding Membership
  - Given: KB, α
  - Decide: KB ⊨ α ? (Is α valid? Is ¬α *contradictory*, *i.e.*, unsatisfiable?)
- Procedure
  - Test whether KB ∪ {¬α} $\vdash_{RESOLUTION} \perp$
  - Answer <u>YES</u> if it does
- $L_{FOL\text{-}SAT}^{C}$ (written L_{SAT}^{C}) Language of Unsatisfiable Sentences
- Dual Problems $L_{VALID} \cong \overline{L_{SAT}} \Leftrightarrow \overline{L_{SAT}} \le L_{VALID}$ (direct proof) $\wedge$

  $L_{VALID} \le \overline{L_{SAT}}$ (refutation resolution)
- <u>Semi</u>-Decidable: L_{VALID}, $L_{SAT}^{C} \in$ RE \ REC ("Find A Contradiction")
  - Recursive enumerable but not recursive
  - Can return in finite steps and answer YES if $\alpha \in L_{VALID}$ or $\alpha \in L_{SAT}^{C}$
  - Can't return in finite steps and answer NO otherwise

Undecidability of L_{VALID}^{C} & L_{SAT}: Review

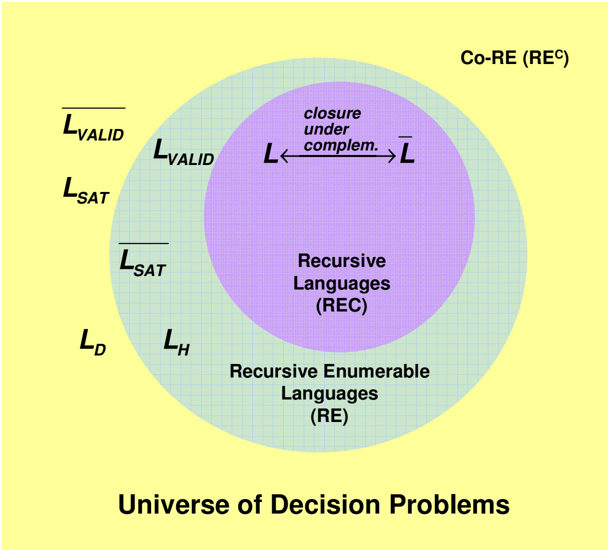
- $L_{FOL\text{-}VALID}^{C}$ (written L_{VALID}^{C}): Language of Non-Valid Sentences
- Deciding Membership
  - Given: KB, α
  - Decide: KB ⊭ α ? (Is there a counterexample to α ? *i.e.*, is ¬α satisfiable?)
- Procedure
  - Test whether KB ∪ { α } $\vdash_{RESOLUTION} \perp$
  - Answer <u>YES</u> if it does NOT
- $L_{FOL\text{-}SAT}$ (written L_{SAT}) Language of Satisfiable Sentences
- Dual Problems $\overline{L_{VALID}} \cong L_{SAT} \Leftrightarrow L_{SAT} \le \overline{L_{VALID}}$ (counterexample) $\wedge$

  $\overline{L_{VALID}} \le L_{SAT}$ (direct proof)
- <u>Un</u>decidable: L_{VALID}^{C}, $L_{SAT} \notin$ RE ("Find A Counterexample")
  - Not recursive enumerable
  - Can return in finite steps and answer NO if $\alpha \notin L_{VALID}^{C}$ or $\alpha \notin L_{SAT}$
  - Can't return in finite steps and answer YES otherwise

Decision Problems: Review

L_H: Halting problem

L_D: Diagonal problem



What Is an Ontology, Anyway?

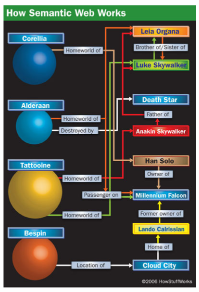
Ontology (information science)

From Wikipedia, the free encyclopedia

This article is about ontology in information science and computer science. For the term in philosophy, see ontology.

In computer science and information science, an **ontology** is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.

In theory, an ontology is a "formal, explicit specification of a shared conceptualisation".[1] An ontology provides a shared vocabulary, which can be used to model a domain — that is, the type of objects and/or concepts that exist, and their properties and relations.[2]



Wilson, T. V. (2006). How Semantic Web Works. http://bit.ly/1AKeOn

Ontologies are used in artificial intelligence, the Semantic Web, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it.

http://en.wikipedia.org/wiki/Ontology_(information_science)

What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals
- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL (often contained in C_2)
    - **Closely related to Propositional Modal, Hybrid & Dynamic Logics**
    - Closely related to Guarded Fragment
  - Provision of inference services
    - Decision procedures for key problems (satisfiability, subsumption, etc)
    - Implemented systems (highly optimised)

 http://bit.ly/10Oh4X

DL Basics

- Concepts (**formulae**)
  - E.g., Person, Doctor, HappyParent, (Doctor ⊔ Lawyer)
- Roles (**modalities**)
  - E.g., hasChild, loves
- Individuals (**nominals**)
  - E.g., John, Mary, Italy
- Operators (for forming concepts and roles) restricted so that:
  - Satisfiability/subsumption is decidable and, *if possible*, of low complexity
  - No need for explicit use of variables
    - Restricted form of ∃ and ∀ (**direct correspondence with ⟨i⟩ and [i]**)
  - Features such as counting (**graded modalities**) succinctly expressed

MANCHESTER 1824

© 2006 I. Horrocks, University of Manchester http://bit.ly/10Oh4X

The DL Family [1]: $\mathcal{ALC}$

- Smallest propositionally closed DL is $\mathcal{ALC}$ (**equivalent to $\mathcal{K}_{(m)}$**)
  - Concepts constructed using booleans
    - $\sqcap, \sqcup, \neg,$

    plus restricted quantifiers
    - $\exists, \forall$
  - Only atomic roles

E.g., Person all of whose children are either Doctors or have a child who is a Doctor:

$$\text{Person} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor})$$

$$\text{Person} \wedge [\text{hasChild}](\text{Doctor} \vee \langle \text{hasChild} \rangle \text{Doctor})$$

The DL Family [2]: $\mathcal{SHOIN}$ & Web Ontology Language

- $\mathcal{S}$ often used for $\mathcal{ALC}$ extended with transitive roles
  - i.e., the union of $\mathcal{K}_{(m)}$ and $\mathbf{K4}_{(m)}$
- Additional letters indicate other extensions, e.g.:
  - $\mathcal{H}$ for role hierarchy (e.g., hasDaughter $\sqsubseteq$ hasChild)
  - $\mathcal{O}$ for **nominals**/singleton classes (e.g., {Italy})
  - $\mathcal{I}$ for inverse roles (**converse modalities**)
  - $\mathcal{Q}$ for qualified number restrictions (**graded modalities**, e.g., $\langle \mathbf{i} \rangle_m \phi$)
  - $\mathcal{N}$ for number restrictions (**graded modalities**, e.g., $\langle \mathbf{i} \rangle_m \top$)
- $\mathcal{S}$ + role hierarchy ($\mathcal{H}$) + nominals ($\mathcal{O}$) + inverse ($\mathcal{I}$) + NR ($\mathcal{N}$) = $\mathcal{SHOIN}$
- $\mathcal{SHOIN}$ is the basis for W3C's OWL Web Ontology Language

DL Knowledge Base

- A TBox is a set of "schema" axioms (sentences), e.g.:

$$\{\text{Doctor} \rightarrow \text{Person},$$
$$\text{HappyParent} \leftrightarrow \text{Person} \wedge [\text{hasChild}](\text{Doctor} \vee \langle\text{hasChild}\rangle\text{Doctor})\}$$

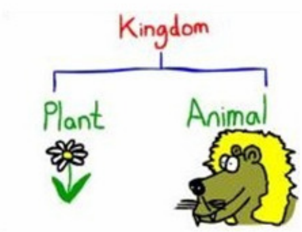
  – i.e., a **background theory** (a set of **non-logical axioms**)
- An ABox is a set of "data" axioms (ground facts), e.g.:

$$\{\text{John} \rightarrow \text{HappyParent},$$
$$\text{John} \rightarrow \langle\text{hasChild}\rangle\text{Mary}\}$$

  – i.e., non-logical axioms including (restricted) use of nominals
- A Knowledge Base (KB) is just a TBox plus an Abox

Adapted from slides © 2006 I. Horrocks, University of Manchester http://bit.ly/10Oh4X

Ontologies and
The Semantic Web (Web 3.0)



Ontologies and OWL

Based on slide © 2006 I. Horrocks, University of Manchester http://bit.ly/10Oh4X

WWW Consortium
Web Ontology Language (OWL)

- Semantic Web led to requirement for a “web ontology language”
- W3C set up Web-Ontology (**WebOnt**) Working Group
  - WebOnt developed **OWL** language
  - OWL based on earlier languages **OIL** and **DAML+OIL**
  - OWL now a W3C **recommendation** (i.e., a standard)
- OIL, DAML+OIL and OWL based on **Description Logics**
  - OWL effectively a “Web-friendly” syntax for $\mathcal{SHOIN}$

Class / Concept Constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	$\neg$Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	$\leqslant$1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	$\geqslant$2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- C is a concept (class); P is a role (property); x_i is an individual/nominal
- XMLS datatypes as well as classes in $\forall$P.C and $\exists$P.C
  - Restricted form of DL concrete domains

Ontology Axioms

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

- OWL ontology equivalent to DL KB (Tbox + Abox)

Why Description Logic?

- **OWL exploits results of 15+ years of DL research**
  - Well defined (model theoretic) **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning algorithms**
  - **Implemented systems** (highly optimised)

Cerebra

FaCT++

Racer

Pellet

Terminology

- Decision Problems: True-False for Membership in Formal Language
  - REC (decidable) vs. RE (semi-decidable OR decidable)
  - Co-RE (undecidable)
  - Russell's Paradox: does the barber shave himself?
- Ontology: Formal, Explicit Specification of Shared Conceptualization
  - Tells what exists (entities, objects)
  - Tells how entities can relate to one another
  - Can be used as basis for reasoning about objects, sets
  - Formalized using logic (e.g., description logic)
- Knowledge Engineering (KE): Process of KR Design, Acquisition
  - Knowledge
    - ⇨ What agents possess (epistemology) that lets them reason
    - ⇨ Basis for rational cognition, action
    - ⇨ Knowledge gain (acquisition, learning): improvement in problem solving
  - Next: more on knowledge acquisition, capture, elicitation
  - Techniques: protocol analysis, subjective probabilities (later)

Summary Points

- Last Class: Resolution Theorem Proving, 9.5 (p. 275-294), R&N 2e
  - Proof example in detail
  - Paramodulation and demodulation
  - Resolution strategies: unit, linear, input, set of support
  - FOL and computability: complements (different difficulty) and duals (same)
- Today: Prolog in Brief, Knowledge Engineering (KE), Ontologies
  - Prolog examples
  - Knowledge engineering
  - Introduction to ontologies
    - ⇨ Ontologies defined
    - ⇨ Ontology design
  - Description logics
    - ⇨ SHOIN
    - ⇨ Web Ontology Language (OWL)
- Next Class: More Ontology Design, KE; Situation Calculus Redux
- Coming Week: Ontologies, Description Logics, Semantic Nets