Declarative Knowledge Processing

Introduction

Magdalena Ortiz

Knowledge Base Systems Group
Institute of Information Systems

ortiz@kr.tuwien.ac.at

10 October 2012
Outline

1. Declarative Knowledge

2. Declarative Knowledge and Reasoning

3. Topics we study in this course
Procedural knowledge, aka imperative knowledge, refers to how to carry out some task or reach some conclusion.

E.g., the knowledge contained in an instruction manual is procedural.

Declarative knowledge, aka descriptive knowledge, states facts that may be employed to carry out some task or reach some conclusion, but does not indicate how to use it.

E.g., the description of a device.
Declarative vs. Procedural Knowledge

**Procedural knowledge**, aka imperative knowledge, refers to how to carry out some task or reach some conclusion.

E.g., the knowledge contained in an instruction manual is procedural.

**Declarative knowledge**, aka descriptive knowledge, states facts that may be employed to carry out some task or reach some conclusion, but does not indicate how to use it.

E.g., the description of a device.

Notes:

- Both kinds of knowledge play an important role in AI.
- An analogous distinction is often done between *procedural* and *declarative* programming languages.
When knowledge is represented declaratively, we need a separate reasoning/inference process to use this knowledge:

- infer implicit knowledge
- use the inferred knowledge to solve problems
- etc.

**Reasoning** (aka inference) is the process that allows us to reach implicit knowledge from **explicitly represented** knowledge.
Declarative Knowledge and Reasoning

When knowledge is represented declaratively, we need a separate reasoning/inference process to use this knowledge:

- infer implicit knowledge
- use the inferred knowledge to solve problems
- etc.

**Reasoning** (aka inference) is the process that allows us to reach implicit knowledge from explicitly represented knowledge.
A formalism for declarative knowledge processing consists of two parts:

1. A language for specifying knowledge
2. Reasoners (aka inference engines) for reasoning about represented knowledge
   - Reasoners solve different reasoning tasks
   - The process is independent of the specific knowledge given to it, and of the domain of application
   - Many reasoners may be available for the same formalism, some may be more adequate for specific settings
Languages for representing declarative knowledge

Languages for representing declarative knowledge must have well defined:

1. **Syntax**: what can we write?
   - Usually given as a set of rules for generating correct statements
   - It must be precisely determined whether an expression is a valid, well-formed statement, or not

2. **Semantics**: what does it mean?
   - The semantics gives meaning to each valid statement
   - It also determines whether a statement is a valid consequence of the represented knowledge

To satisfy this, one usually considers languages based on logic
Examples of formalisms

Formalisms for processing declarative knowledge include:

- **SAT solvers** are used for solving combinatorial problems, and are based on classical propositional logic.

- **Answer Set Programming** is a declarative logic programming formalism used for solving hard problems, including combinatorial problems too hard to be solved with SAT solvers.

- There are several formalisms for processing ontological knowledge like:
  - Ontology Web Languages
  - Description Logics
  - RDF
  - F-logic
  - Rule based formalisms based on Datalog and its extensions
We will survey some of the main formalisms for processing declarative knowledge.

In particular, we focus on:

1. The expressive Description Logics that are used for ontological reasoning
2. The lightweight Description Logics that are used for Ontology Based Data Access
3. Datalog and its extensions
4. Answer Set Programming, one of the most prominent extensions of Datalog
If you have not done so yet, please register on TISS!
Organizational issues

If you have not done so yet, please register on TISS!

- One session of 2h 15’ (or 2h 30’ with break) per week
- Tentative schedule: Wednesday, 9:15-11:30
- 9 sessions, plus one shorter session for discussion before the exam
- Evaluation: oral exam at the end of the course
- The whole course (including exam) will be concluded before the Christmas holidays

The exam need not be based exclusively on the slides, but may include additional material discussed in class or assigned for independent reading.
Organizational issues

If you have not done so yet, please register on TISS!
If you have not done so yet, please register on TISS!

- One session of 2h 15’ (or 2h 30’ with break) per week
- 9 sessions, plus one shorter session for discussion before the exam
- Evaluation: oral exam at the end of the course
- The whole course (including exam) will be concluded before the Christmas holidays
- Tentative schedule: Wednesday, 9:15-11:30

The exam need not be based exclusively on the slides, but may include additional material discussed in class or assigned for independent reading.
## Tentative schedule in detail

(still subject to change!)

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.2012</td>
<td>Intro, Recap of complexity theory and propositional logic</td>
</tr>
<tr>
<td>17.10.2012</td>
<td>Description Logics 1: basics, reasoning problems</td>
</tr>
<tr>
<td>31.10.2012</td>
<td>Description Logics 2: ALC</td>
</tr>
<tr>
<td>07.11.2012</td>
<td>Description Logics 3: Tableaux for ALC</td>
</tr>
<tr>
<td>14.11.2012</td>
<td>Description Logics 4: More expressive DLs, the OWL languages</td>
</tr>
<tr>
<td>21.11.2012</td>
<td>Description Logics for OBDA 1: DL-Lite and query rewriting</td>
</tr>
<tr>
<td>28.11.2012</td>
<td>Description Logics for OBDA 2: Beyond DL-Lite</td>
</tr>
<tr>
<td>05.12.2012</td>
<td>Datalog and its extensions</td>
</tr>
<tr>
<td>12.12.2012</td>
<td>Answer set programming</td>
</tr>
<tr>
<td>19.12.2012</td>
<td>Questions for the exam and discussion</td>
</tr>
</tbody>
</table>