Semantic Web Foundations

Part 1: Modeling in Description Logic

Peter Radics

Goal

- Goal of presentation:
  - Introduce building blocks of Description Logic
  - Provide starting point for modeling in Description Logic
  - Take away fear of difficult-sounding domain

History

- First knowledge representation systems in the 1970s
  - Focused on high level descriptions of the world for intelligent applications
- Approaches roughly divided into:
  - Logic-based formalisms
  - Non-logic-based representations

History (cont'd)

- Non-logic-based representations
  - Frames
  - Semantic Networks
- Rely on network-based representation structures
  - Nodes
  - Links

History (cont'd)

- However:
  - "...early Semantic Networks suffered from the drawback that they did not have clear semantics."

History (cont'd)

- Core features of frames and semantic networks and first-order logic can provide clear semantics
- → Description Logic (DL)

Aside: First-Order Logic

- Example of a typical statement:

$$C \equiv \square x \; \square y \; \square R \, \square a,x \, \square\square \; R \, \square a,y \, \square\square \; M \; \square x \, \square\square \; \neg M \; \square y \, \square\square$$

  - Hard to read and interpret by non-mathematicians

DL: Concepts

- DL is “object-centered modeling language”

- Concepts (Classes, Nodes)
  - Collections of Individuals with same properties
  - Two default concepts (for reasoning):
    - Thing
    - Nothing
  - Modeled as **unary symbols** in first-order logic

DL: Concepts (cont'd)

- Concept definition:
  - Provides both necessary and sufficient information for classifying individual
  - Establishes logical equivalence
  - Acyclic
- Classification basic task in constructing terminology

DL: Relationships

- Relationships (Links, Slots, Roles)
  - Subsumption (is-a relationship)
    - Relationship shared with many other modeling languages (e.g. Entity-Relationship diagrams)
    - Used for building taxonomy of classes
  - However, DL allows for arbitrary (binary) relationships
  - Modeled as **binary symbols** in first-order logic

DL: T-Box

- Together, concepts and relationships form terminology (T-Box)
- Terminology models intensional knowledge (i.e. general domain knowledge)

DL: Individuals

- Individuals
  - Instances (members) of classes
  - Convey assertional/extensional knowledge (i.e. problem specific knowledge about a domain)
  - Form A-Box

Aside: Expressiveness

- Do we have enough to define:
  - Concept “Male students”?
  - Concept “Friends and family”?
  - Concept “Non-smokers”?
  - Concept “Parents?”
  - Concept “Parents of only girls”?
  - Concept “Parents with three children”?
- →Additional building blocks needed

DL: Additional building blocks

- Added to increase expressiveness
  - Intersection of concepts (logical and)
    - Allows for:
      - MaleStudent = Male and Student
  - Union of concepts (logical or)
    - Allows for:
      - FriendsAndFamily = Friends or Family
  - Complement of concepts (logical not)
    - Allows for:
      - NonSmoker = not Smoker

DL: Additional building blocks

- Existential quantification
  - Allows for:
    - Parents = exists hasChild (a, x)
- Universal quantification
  - Allows for:
    - ParentsOfOnlyGirls = for all hasChild (a,x) Female(x)
- Cardinality restriction
  - Allows for
    - ParentsWithThreeChildren = (>=3 hasChild) and (<=3 hasChild)

Example

- First-Order Logic example:

$$C \equiv \exists x \, \exists y \, (R(a,x) \wedge R(a,y) \wedge M(x) \wedge \neg M(y))$$

- Becomes:
  - ParentWithSonAndDaughter = hasChild.x and hasChild.y and x.Male and y.Female

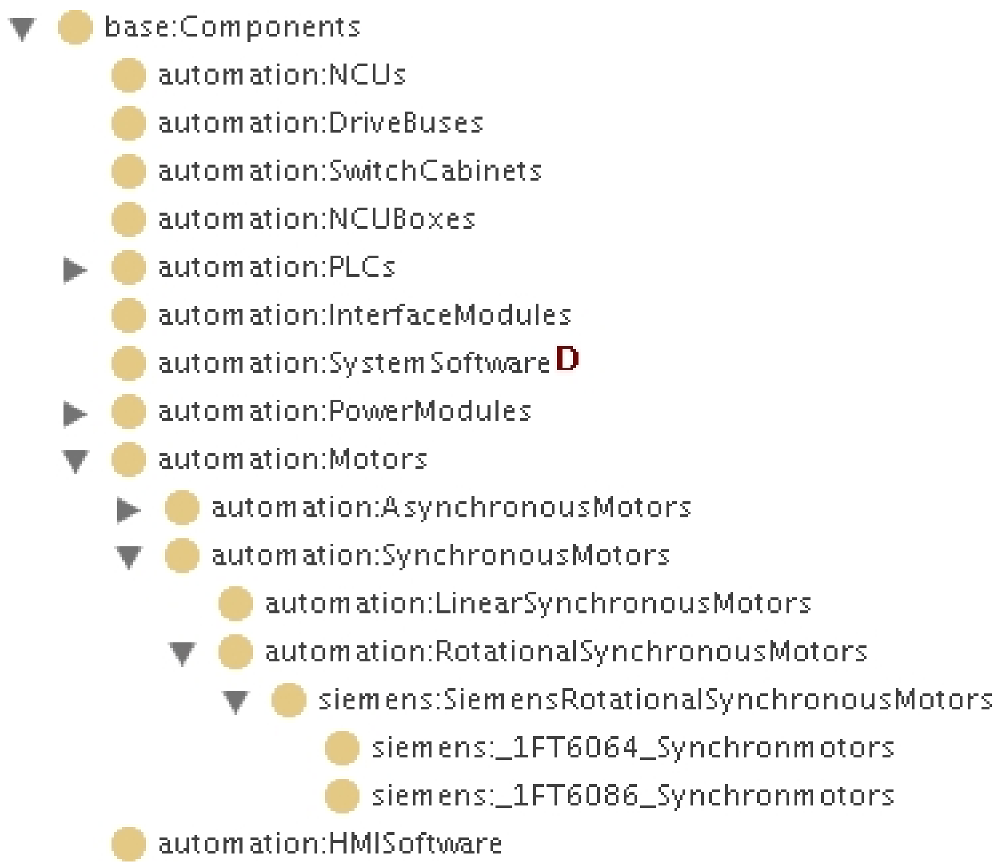
DL: Modeling

- Knowledge base should clearly characterize the question it can answer.
- Model hast to be complete before reasoning can be applied.
- Expressiveness of DL language influences complexity of reasoning

DL: Making it user-friendly

- Two approaches:
  - Providing syntax that is closer to natural language
  - Providing graphical user interface for specifying relationships

Real-World examples



```
▼ ● base:Components
      ● automation:NCUs
      ● automation:DriveBuses
      ● automation:SwitchCabinets
      ● automation:NCUBoxes
    ▶ ● automation:PLCs
      ● automation:InterfaceModules
      ● automation:SystemSoftware D
    ▶ ● automation:PowerModules
    ▼ ● automation:Motors
        ▶ ● automation:AsynchronousMotors
        ▼ ● automation:SynchronousMotors
              ● automation:LinearSynchronousMotors
            ▼ ● automation:RotationalSynchronousMotors
                ▼ ● siemens:SiemensRotationalSynchronousMotors
                      ● siemens:_1FT6064_Synchronmotors
                      ● siemens:_1FT6086_Synchronmotors
      ● automation:HMISoftware
```

Real world examples

```
<base:BoltedClampConnection
  rdf:ID="SmartflexClutchSize3_RotatingBall
  Screw">
<base:hasPosition
  rdf:resource="Pos_Connection_Clutch_Ba
  llScrew"/>
<base:isConnectionOfObject1
  rdf:resource="#SmartflexClutchSize3"/>
<base:isConnectionOfObject2
  rdf:resource="#RotatingBallScrew"/>
```

Conclusion

- Description Logic is not “scary”
- Allows modeling of real world knowledge in vocabulary similar to natural language

Recommended Read

- Noy, McGuinnes: “Ontology Development 101: A Guide to Creating Your First Ontology”, Tech Report, Knowledge Systems Laboratory, Stanford University, 2001

Discussion

- How does modeling in Description Logic apply to Usable Security?
- What are potential benefits?
- What are potential downfalls?

Outlook: Reasoning

- Open world assumption of DL
  - Example:
    - hasChild (Iokaste, Oedipus)
    - hasChild (Iokaste, Polyneikes)
    - hasChild (Oedipus, Polyneikes)
    - hasChild (Polyneikes, Thersandros)
    - Patricide (Oedipus)

      ¬Patricide (Thersandros)

  - Question: Does Iokaste have a child that is a patricide and that itself has a child who is not a