Semantic Networks and Description Logics

Description Logics in the Semantic Web

Knowledge Representation and Reasoning

January 9, 2005

Description Logics in the Semantic Web

Ontology

- Ontology is a term borrowed from Philosophy.
- An ontology defines the terms used to describe and represent an area of knowledge.
- Ontologies are used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.).
- Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them.
- They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge *reusable*.

An Ontology Language for the Web

- Why an ontology language for the web?
- Berners-Lee, Hendler, and Larissa stated in the *Scientific American*:

  For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Artificial Intelligence researchers have studied such systems since long before the Web was developed. Knowledge representation, as this technology is often called, is currently in a state comparable to that of hypertext before the advent of the Web: it is clearly a good idea, and some very nice demonstrations exist, but it has not yet changed the world. It contains the seeds of important applications, but to realize its full potential it must be linked into a single global system.

OWL - The Web Ontology Language

- OWL is an XML language
- OWL extends RDFS (Resource Description Framework – Schema)
- OWL has a precise formal semantics based on Description Logics
- There are three versions
  - OWL *Lite* – small subset (corresponds to DL $\mathcal{SHIF}(D)$)
    - (D) meaning that there can be an extra data domain
  - OWL *DL* – full language with a number of semantic restrictions (corresponds to DL $\mathcal{SHIQ}(D)$)
  - OWL *Full* – full language, which has however undecidable reasoning problems
- Reasoning services are provided *off-line* – currently

Transitive Closure and Transitive Roles

- Often *transitivity* of a role is required. For example the role *part-of* is usually transitive.
- One possibility is to add a role-forming operator *transitive closure*:

$$(r^+)^{\mathcal{I}} = \bigcup_{i \in \mathbb{N}} (r^{\mathcal{I}})^i$$

$$(r^{\mathcal{I}})^i = \begin{cases} r^{\mathcal{I}}, & \text{if } i = 1 \\ r^{\mathcal{I}} \circ (r^{\mathcal{I}})^{i-1}, & \text{otherwise} \end{cases}$$

- $\mathcal{ALC}$ + transitive closure: $\mathcal{ALC}_+$
- Another possibility is the introduction of a special type of roles that are by definition transitive.
- $\mathcal{ALC}$ + transitive roles: $\mathcal{ALC}_{R^+}$
- Transitive roles are algorithmically easier than transitive closure, although computational complexity does not seem to make a difference in most cases.

The $\mathcal{S}$-Family

- $\mathcal{ALC}_{R^+}$ is also called $\mathcal{S}$ due to its relationship to the multi-modal logic $\mathbf{S4}_{(\mathbf{m})}$.
- Often one wants role inclusions (a *role hierarchy*), i.e., a set of statements: $r \sqsubseteq s$.
- The language $\mathcal{SH}$
- *Inverse roles* are also often needed
- The languages $\mathcal{SI}$ and $\mathcal{SHI}$
- *Qualified number restrictions* $(\leq n\, r.C)$ can also be added, but only on **simple roles**, roles that are neither transitive nor have a transitive sub-role
- The languages $\mathcal{S} \ldots \mathcal{Q}$
- $\mathcal{N}$ is the simplification to simple number restrictions $(\leq n\, r)$, and $\mathcal{F}$ to functional number restrictions $(\leq 1\, r)$.
- In what follows, we focus on $\mathcal{SHIF}$.

$\mathcal{SHIF}$-Terminologies

- Set of concept names $\mathbf{A}$, set of role names $\mathbf{R}$, and a subset $\mathbf{R}_+ \subseteq \mathbf{R}$ of transitive roles
- For each role $r \in \mathbf{R}$, there exists the inverse role $r^- \in \mathbf{R}$, whereby enforce that on a syntactic level $r^{--} \equiv r$
- The $\mathcal{ALC}$ concept forming operators + $(\leq 1\, r)$
- A role hierarchy $\mathcal{R}$, which is a set of role inclusion statements $r \sqsubseteq s$
- A general terminology $\mathcal{T}$, which is a set of concept inclusion statements $C \sqsubseteq D$ (note: no restriction concerning the left hand side, cyclicity, or double definitions)
- Models, satisfiability, subsumption, etc. is defined in the same way as for "simple" $\mathcal{ALC}$-terminologies.

Getting Rid of a $\mathcal{SHIF}$-TBox

- Similar to $\mathcal{ALC}$, we want to get rid of the TBox when determining subsumption
- However, normalization and unfolding does not work
- The technique used in case of $\mathcal{SHIF}$ and similar DLs is called *internalization*
- Idea: Encode the constraints specified by the inclusion statements in $\mathcal{T}$ in a concept expression $C_{\mathcal{T}}$ and require that all domain elements are instances of $C_{\mathcal{T}}$
- This requirement can be enforced by introducing a *quasi-universal*, transitive role u and stating $\forall u.C_{\mathcal{T}}$.

Internalization (1)

Lemma (Internalization)

Let $\mathcal{R}$ be a $\mathcal{SHIF}$ role hierarchy, $\mathcal{T}$ be a $\mathcal{SHIF}$ terminology, C be a $\mathcal{SHIF}$ concept, and let

$$C_{\mathcal{T}} = \sqcap_{(C_i \sqsubseteq D_i) \in \mathcal{T}} (\neg C_i \sqcup D_i).$$

Let u be a transitive role not appearing in $\mathcal{R}$, $\mathcal{T}$, or C. We set

$$\mathcal{R}_u = \mathcal{R} \cup \{r \sqsubseteq u, r^- \sqsubseteq u \mid r \text{ occurs in } \mathcal{R}, \mathcal{T}, \text{ or } C\}.$$

Then C is satisfiable w.r.t $\mathcal{T}$ and $\mathcal{R}$ iff $C \sqcap C_{\mathcal{T}} \sqcap \forall u.C_{\mathcal{T}}$ is satisfiable w.r.t $\mathcal{R}_u$.

Internalization (2)

Proof.

$\Rightarrow$: Assume C is satisfiable in $\mathcal{R}$ and $\mathcal{T}$. Then there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{R}$ over the domain $\mathcal{D}$ s.t. there exists $d \in C^{\mathcal{I}}$. Let u be the *universal role*, i.e., $u^{\mathcal{I}} = \mathcal{D} \times \mathcal{D}$. Obviously $d \in C^{\mathcal{I}}$. Further for all $x \in \mathcal{D}$: if $x \in C_i^{\mathcal{I}}$ then $x \in D_i^{\mathcal{I}}$ for all $(C_i \sqsubseteq D_i) \in \mathcal{T}$. This implies $x \in (\neg C_i \sqcup D_i)^{\mathcal{I}}$. Hence, we have $d \in C_{\mathcal{T}}^{\mathcal{I}}$. Since we also have $x \in C_{\mathcal{T}}^{\mathcal{I}}$ for all $x \in \mathcal{D}$ and since u is the universal role, we have also $d \in (\forall u.C_{\mathcal{T}})^{\mathcal{I}}$, i.e., altogether we get

$$d \in (C \sqcap C_{\mathcal{T}} \sqcap \forall u.C_{\mathcal{T}})^{\mathcal{I}},$$

which means $(C \sqcap C_{\mathcal{T}} \sqcap \forall u.C_{\mathcal{T}})$ is satisfiable w.r.t. $\mathcal{R}_u$.

$\Leftarrow$: Assume $d \in (C \sqcap C_{\mathcal{T}} \sqcap \forall u.C_{\mathcal{T}})^{\mathcal{I}}$ for some model $\mathcal{I}$ of $\mathcal{R}_u$ over the domain $\mathcal{D}$. We can constrain ourselves to the sub-domain consisting of the elements x such that $(d,x) \in u^{\mathcal{I}}$. Note that in this domain all elements satisfy the inclusion statements (because of $\forall u.C_{\mathcal{T}}$). Hence, it is a model of $\mathcal{T}$ and $\mathcal{R}$ and $d \in C^{\mathcal{I}}$ by assumption, i.e., C is satisfiable w.r.t. $\mathcal{T}$ and $\mathcal{R}$. □

Extending the Tableau Algorithm

- For tableau algorithms one has to show that
  1. a tableau that does not contain a contradiction corresponds to a model
  2. all ways to construct a model are systematically tried
  3. the algorithm always terminates
- Termination is easy for $\mathcal{ALC}$ because the concept expressions are decomposed into smaller and smaller expressions
- This is not true for transitive roles anymore.

Blocking for $\mathcal{S}$ and $\mathcal{SH}$

- Let us assume that the only additional rule is:

  if $xry, yrz \in S$ then xrz should be added to S
- Consider the system $S = \{x : C, x : \exists r.C, x : \forall r.(\exists r.C)\}$, where r is transitive
- Expanding this would result in:

$$S \cup \{xry, y : C, y : \exists r.C\}$$
$$S \cup \{xry, y : C, y : \exists r.C, yrz, xrz, z : C, z : \exists r.C\}$$

- Further expansion can be **blocked**: We say *x blocks y*
- In this case, we *identify* the old (blocking) and the new (blocked) node and thus form a cycle!
- Blocking can always be done – if the new variable (y) has a subset of the constraints of the generating variable (x)
- Note: Transitive closure roles could be handled similarly, but one gets non-determinism and "bad" cycles have to be recognized, i.e., it is significantly more complicated.

Blocking in the Presence of Transitive Closure

- One can can use the same blocking condition as in the case of transitive roles – in principle
- $\exists r^+.C$ has to be *non-deterministically expanded* into
  1. $\exists r.C$ and
  2. $\exists r.\exists r^+.C$
- When checking for a blocking condition, one has to check whether the cycle is *good*. Example:

$$D \;=\; \exists r^+.A \sqcap \forall r^+.\neg A \sqcap \neg A$$

- A run of the tableau algorithm might generate:

$$S_1 \;=\; \{x : \exists r^+.A, x : \exists r.(\exists r^+.A), x : \forall r^+.\neg A, x : \neg A\}$$
$$S_2 \;=\; S_1 \cup \{xry, y : \exists r^+.A, y : \neg A\}$$

- This cycle is not *good* because we never had the expansion of $\exists r^+.A$ to $\exists r.A$.
- Before blocking, one has to check whether the cycle is "good."

Blocking in $\mathcal{SHI}$

With inverse roles, things are a bit more complicated:

- Blocking can only be allowed when we have equal sets of constraints (because roles are "bi-directional" now)
- Assume that a successor of $x : C, x : \exists r.C, x : \forall r.(\exists r.C)$ is blocked by a node containing $\forall r^-.\neg C$. Then closing the cycle leads to an inconsistency.
- Furthermore, expansions anywhere in the node can lead to propagations to the blocked or blocking node
- *Dynamic blocking*: Blocks can be established and dynamically broken!
- It becomes much more difficult to show termination.

Blocking in $\mathcal{SHIF}$

- With functional restrictions and inverse roles, we can force models to become *infinite*.
- Consider:

$$\neg C \sqcap \exists f^-.(C \sqcap (\leq 1\, f)) \sqcap \forall r^-.(\exists f^-.(C \sqcap (\leq 1\, f))),$$

  where r is transitive and $f \sqsubseteq r$.
- This concept has only infinite models, namely, sequences of individuals related by f^- and all satisfying $C \sqcap \exists f^-.C$
- One cannot close the cycle, since then the whole sequence would have to collapse into one node because of the functional restriction
- More sophisticated *pair-wise blocking* strategy necessary.

Simple and General Roles

- When adding $\mathcal{Q}$ ($\mathcal{N}$ or $\mathcal{F}$), we required that the roles to be restricted are *simple*, i.e., not transitive and there is no sub-role that is transitive
- Would constrain the number of possible (indirectly) reachable successors to be a particular number, which sounds funny
- Leads indeed to undecidability of the satisfiability problem for *unrestricted* $\mathcal{SHN}$ and, of course, $\mathcal{SHQ}$.
- These unrestricted variants are called $\mathcal{SHQ}^+$, $\mathcal{SHN}^+$, and $\mathcal{SHF}^+$.
- A reduction from the *domino problem*, which is undecidable, to satisfiability in $\mathcal{SHN}^+$ is used to show undecidability.

The Domino Problem

Definition (Domino Problem)

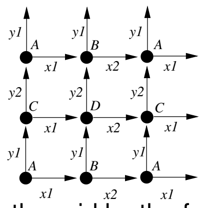
A domino system $\mathcal{D} = \langle D, H, V \rangle$ consists of a non-empty set of domino types $D = \{D_1, \ldots, D_n\}$, and sets of horizontally and vertically matching pairs $H \subseteq D \times D$ and $V \subseteq D \times D$. The *domino problem* is to determine if, for a given $\mathcal{D}$, there exists a tiling of an $\mathbb{N} \times \mathbb{N}$ grid such that each point of the grid is covered with a domino type in D and all horizontally and vertically adjacent pairs of domino types are in H and V respectively, i.e., a mapping $t : \mathbb{N} \times \mathbb{N} \rightarrow D$ such that for all $m, n \in \mathbb{N}$, $\langle t(m,n), t(m+1,n) \rangle \in H$ and $\langle t(m,n), t(m,n+1) \rangle \in V$.

Theorem (Undecidability)

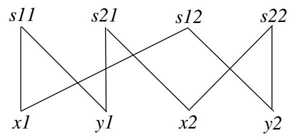
The domino problem is undecidable

Constructing the Domino Grid

We want a grid as follows:



We use the following role hierarchy (sij transitive):



We force the grid by the following concept inclusion statements:

$$A \sqsubseteq \neg B \sqcap \neg C \sqcap \neg D \sqcap \exists x1.B \sqcap \exists y1.C \sqcap (\leq 3 s11)$$
$$B \sqsubseteq \neg A \sqcap \neg C \sqcap \neg D \sqcap \exists x2.A \sqcap \exists y1.D \sqcap (\leq 3 s21)$$
$$C \sqsubseteq \neg A \sqcap \neg B \sqcap \neg D \sqcap \exists x1.D \sqcap \exists y2.A \sqcap (\leq 3 s12)$$
$$D \sqsubseteq \neg A \sqcap \neg B \sqcap \neg C \sqcap \exists x2.C \sqcap \exists y2.B \sqcap (\leq 3 s22)$$

Undecidability of $\mathcal{SHN}^+$

- We can uniquely specify the *grid structure*
- Now, one only has to encode the *domino type constraints*, which is a piece of cake

Theorem

Satisfiability and subsumption in $\mathcal{SHN}^+$ are undecidable.

Note: The border between decidable (EXPTIME-complete) and undecidable DLs is quite thin!

Conclusion & Outlook

- There is a strong need for *ontologies* on the World Wide Web
- OWL (Web Ontology Language) is designed for this purpose
- It extends RDFS
- The *OWL Lite* and *OWL DL* fragments are based on DLs
- *OWL Full* goes beyond it and is undecidable, even $\mathcal{DHN}^+$ is already undecidable
- Even the restricted fragments require sophisticated techniques such as *internalization* and *blocking*
- Efficient implementation techniques, which we have not seen yet, have been developed
- In many practical cases, DL reasoners are efficient enough to classify large KBs

Literature

- W3C documents: `http://www.w3c.org/2004/OWL/`
- I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239-264, May 2000.
- I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, **9**(3): 385–410, 1999.