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.

Description Logics in the Semantic Web

OWL – Web Ontology Language

The $S$-family

$SHIF$-Terminologies and Subsumption

Tableau Reasoning

Undecidability of Unrestricted $SHN$

Conclusion
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:
  
  $$ (r^+)^I = \bigcup_{i \in \mathbb{N}} (r^I)^i $$
  
  $$(r^I)^i = \begin{cases} r^I, & \text{if } i = 1 \\ r^I \circ (r^I)^{i-1}, & \text{otherwise} \end{cases}$$

- Another possibility is the introduction of a special type of roles that are by definition transitive.
- Transitive roles are algorithmically easier than transitive closure, although computational complexity does not seem to make a difference in most cases.

The S-family

- $\mathcal{ALC}^+_R$: is also called $S$ due to its relationship to the multi-modal logic $S4_{(m)}$.
- 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 ($\leq n \cdot r.C$) can also be added, but only on simple roles, roles that are neither transitive nor have a transitive sub-role
- The languages $S\ldots Q$
- $\mathcal{N}$ is the simplification to simple number restrictions ($\leq n \cdot r$), and $\mathcal{F}$ to functional number restrictions ($\leq 1 \cdot r$).
- In what follows, we focus on $\mathcal{SHIF}$. 

$\mathcal{SHIF}$-Terminologies

- Set of concept names $A$, set of role names $R$, and a subset $R_+ \subseteq R$ of transitive roles
- For each role $r \in R$, there exists the inverse role $r^- \in R$, whereby we on a syntactical level that $r^- \equiv r$
- The $\mathcal{ALC}^+$ concept forming operators $+ (\leq 1 \cdot r)$
- A role hierarchy $R$, which is a set of role inclusion statements $r \sqsubseteq s$
- A general terminology $T$, 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 $SHIF$-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 $SHIF$ and similar DLs is called **internalization**
- Idea: Encode the constraints specified by the inclusion statements in $T$ in a concept expression $C_T$ and require that all domain elements are instances of $C_T$
- This requirement can be enforced by introducing a quasi-universal, transitive role $u$ and stating $\forall u. C_T$

Internalization (1)

**Lemma (Internalization)**
Let $R$ be a $SHIF$ role hierarchy, $T$ be a $SHIF$ terminology, $C$ be a $SHIF$ concept, and let

$$C_T = \bigcap_{(C_i \sqsubseteq D_i) \in T} \neg C_i \sqcup D_i.$$ 

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

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

Then $C$ is satisfiable w.r.t $T$ and $R$ iff $C \sqcap C_T \sqcap \forall u.C_T$ is satisfiable w.r.t $R_u$.

Internalization (2)

**Proof.**

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

which means $(C \sqcap C_T \sqcap \forall u.C_T)$ is satisfiable w.r.t. $R_u$.

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

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 $S$ and $SH$

- Let us assume that the only additional rule is:
  
  \[ \text{if } xry, yrz \in S \text{ then } xrz \text{ should be added to } S \]

- 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:
  
  \[
  \begin{align*}
  S &\cup \{ xry, y : C, y : \exists r.C \} \\
  S &\cup \{ xry, y : C, y : \exists r.C, yrz, z : C, z : \exists r.C \}
  \end{align*}
  \]

- Further expansion can be blocked: We say \( x \) blocks \( y \)

\[ \rightsquigarrow \]

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 \( x \)

\[ \rightarrow \]

Note: Transitive closure roles could be handled similarly, but one gets non-determinism and “bad” cycles to be recognized, i.e., it is significantly more complicated.

---

Blocking in the Presence of Transitive Closure

- One 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. $\exists r.C$
  2. $\exists r.\exists r+.C$

- When checking for a blocking condition, one has to check whether the cycle is good. Example:

\[ D = \exists r+.A \cap \forall r+.\neg A \cap \neg A \]

- 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 $\exists r.A$.

\[ \rightsquigarrow \]

Before blocking, one has to check whether the cycle is “good.”

---

Blocking in \( SHI \)

- 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

\[ \rightsquigarrow \]

**Dynamic blocking**: Blocks can be established and dynamically broken!

\[ \rightarrow \]

It becomes much more difficult to show termination.

---

Blocking in \( SHIF \)

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

- Consider:

\[ \neg C \cap \exists f^-.(C \cap (\leq 1 f)) \cap \forall r^-.(\exists f^-.(C \cap (\leq 1 f))), \]

where $r$ is transitive and $f \subseteq r$.

- This concept has only infinite models, namely, sequences of individuals related by $f^-$ and all satisfying $C \cap \exists f^- . C$

\[ \rightsquigarrow \]

One cannot close the cycle, since then the whole sequence would have to collapse into one node because of the functional restriction

\[ \rightarrow \]

More sophisticated pair-wise blocking strategy necessary.
Simple and General Roles

- When adding \( Q \) (\( N \) or \( F \)), we required that the roles to be restricted are \textit{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 \( SHN \) and, of course, \( SHQ \).
- These unrestricted variants are called \( SHQ^+ \), \( SHN^+ \), and \( SHF^+ \).

- A reduction from the \textit{domino problem}, which is undecidable, to satisfiability in \( SHN^+ \) is used to show undecidability.

The Domino Problem

Definition (Domino Problem)
A domino system \( D = \langle D, H, V \rangle \) 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 \textit{domino problem} is to determine if, for a given \( D \), there exists a tiling of an \( N \times N \) 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 \( t : N \times N \rightarrow D \) such that for all \( m, n \in N \), \((t(m, n), t(m + 1, n)) \in H \) and \((t(m, n), t(m, n + 1)) \in V \).

Theorem (Undecidability)
The domino problem is undecidable

Constructing the Domino Grid

We want a grid as follows:

\[
\begin{array}{c|c|c|c}
\hline
\text{A} & \text{B} & \text{C} \\
\hline
\text{x1} & \text{x2} & \text{x1} \\
\hline
\text{y1} & \text{y2} & \text{y1} \\
\hline
\text{y2} & \text{y1} & \text{y2} \\
\hline
\end{array}
\]

We use the following role hierarchy (\( sij \) transitive):

\[
\begin{align*}
\text{s11} & \rightarrow \text{s21} \\
\text{s12} & \rightarrow \text{s22} \\
\text{x1} & \rightarrow \text{y1} \\
\text{x2} & \rightarrow \text{y2}
\end{align*}
\]

We force the grid by the following concept inclusion statements:

\[
\begin{align*}
A \subseteq \neg B \land \neg C \lor \neg D \\
B \subseteq \neg A \land \neg C \lor \neg D \\
C \subseteq \neg A \land \neg B \lor \neg D \\
D \subseteq \neg A \land \neg B \land \neg C
\end{align*}
\]

Undecidability of \( SHN^+ \)

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

Theorem

\textit{Satisfiability and subsumption in } \( SHN^+ \text{ are undecidable.} \)

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 $\mathcal{D\mathcal{H}\mathcal{N}}^+ \text{ is already undecidable}$
- 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/