

Practical Reasoning for the Semantic Web

Day 2

Stefan Schlobach & Heiner Stuckenschmidt

August 16, 2005

<http://www.few.vu.nl/~schlobac/>

What we do Today

- ▶ The SW and OWL
- ▶ Introducing Description Logics
 - Historical background
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
 - Structural subsumption
 - Tableau calculi
- ▶ Implementation and Optimisation issues

What we do Today

- ▶ The SW and OWL
- ▶ Introducing Description Logics
 - Historical background
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
 - Structural subsumption
 - Tableau calculi
- ▶ Implementation and Optimisation issues

The Semantic Web

Most existing Web resources are only understandable by humans

- ▶ Markup (HTML) provides rendering information
- ▶ Textual/graphical information for human consumption
- ▶ Semantic Web aims at machine understandability
 - *Semantic* markup will be added to web resources
 - Markup will use *Ontologies* for shared understanding
 - Requirements for OWL ontology language
 - Should extend existing Web standards (XML, RDF, RDFS)
 - OWL

Semantic Information

- ▶ Current web-pages are used for layout and routine processing such as links, headers or programs
- ▶ but no semantics that could be understood by a computer.
- ▶ The computer does not know anything about art, e.g.
 - that napoleon is a person, but not a painter
 - that parijs is a city
 - that parijs and Paris∈France are the same object.
- ▶ More than keyword search is needed
- ▶ We want a WWW that formalizes semantic information so that it can be used by intelligent *agents*.



Classical logical reasoning in the SW

- ▶ Ontology modeling
 - correctness of definitions
 - correctness of hierarchy
- ▶ Give me all instances belonging to a particular class
 - *give me all paintings from Dutch painter, which lived abroad*
- ▶ Give me the most specific class for an instance
 - cataloging paintings according to content
 - classify patients w.r.t. their medical diagnosis

Correct reasoning can be **very** important (e.g. medical domain)

Ontologies

- ▶ In Philosophy: an ontology is a theory about the nature of existence, of what types of things exist.
- ▶ In AI: Document or file that formally defines relations among terms and individuals.
 - Define classes and
 - relation among them.
 - Plus: inferences.
- ▶ The W3C adopted Ontology language is called OWL.

Why OWL ?

- ▶ OWL combines
 - Description Logic, but adds
 - references of names of objects and classes
 - this allows to combine different ontologies
 - XML schema datatypes
 - the connection to the WWW
- Frame paradigm, but adds
 - formal semantics
- RDF, but adds
 - complex language features
 - while maintaining upwards compatibility
- ▶ Tries to combine expressiveness and decidability

Problems with OWL

As OWL has to integrate features from different paradigms a number of problems had (and still have) to be solved:

- ▶ Syntactic Problems
 - very verbose (and ugly) syntax
 - OWL constructs have to be encoded as triples
 - Therefore, there is an abstract syntax (closer to frames and DL)
- ▶ Semantic Problems
 - OWL semantics must be fully compatible with RDF and RDFS.
 - which is difficult as RDF requires explicit information on syntax.
 - OWL needs to be very expressive.
- ▶ Computational Problems
 - OWL is rich in possible inferences
 - Including all the language desires would render the language undecidable
 - and even the decidable fragments are computationally very bad.

Versions of OWL

To appease different communities, there are three versions of OWL.

- ▶ OWL lite:
 - Simple ontology language with efficient reasoning (e.g. RACER)
 - doesn't allow classes as instances (to ensure decidability)
- ▶ OWL DL:
 - Expressive ontology language with decidable inference
 - allows data-typing, RDF URI references, including names from RDF, RDFS and XML Schema data-types.
- ▶ OWL full:
 - Very expressive ontology language
 - upwards compatible with RDF
 - Entailment is undecidable.

What we do Today

- ▶ The SW and OWL
- ▶ Introducing **Description Logics**
 - **Historical background**
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
 - Structural subsumption
 - Tableau calculi
- ▶ Implementation and Optimisation issues

Description Logic: Semantics

- ▶ Interpretations are pairs (Δ, \mathcal{I}) , with a universe Δ and a mapping \mathcal{I} from
 - concept names to subsets of Δ
 - role names to binary relations

- ▶ Booleans: $C \sqcap D, (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $C \sqcup D, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $\neg C, (\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$

Qualified quantification:

$$\forall R.C = \{x \in \Delta \mid \forall y \in \Delta : R^{\mathcal{I}}(x, y) \rightarrow y \in C^{\mathcal{I}}\}$$

$$\exists R.C = \{x \in \Delta \mid \exists y \in \Delta : R^{\mathcal{I}}(x, y) \& y \in C^{\mathcal{I}}\}$$

Concept Reasoning

Based on these semantics there are two basic reasoning services:

- ▶ **Concept satisfiability**, $\models C \neq \perp$.
- Check whether for some interpretation \mathcal{I} we have $C^{\mathcal{I}} \neq \emptyset$.
- $\models \forall \text{creates. Sculpture} \sqcap \exists \text{creates. (Artwork} \sqcap \neg \text{Sculpture)} = \perp$.
- ▶ **Concept subsumption**, $\models C_1 \sqsubseteq C_2$.
- Check whether for all interpretations \mathcal{I} we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- $\forall \text{creates. Painting} \sqcap \exists \text{creates. T} \sqsubseteq \exists \text{creates. Painting}$.

The family of Description Logics

- ▶ Historically there were three phases:
 - very expressive DLs, but undecidable: systems e.g. Loom
 - tractable DLs, but very inexpressive languages
 - expressive and decidable DLs, optimised systems RACER or FaCT
- ▶ We look at the latter two:
 - e.g.: \mathcal{FL}^-
 - conjunction $C \sqcap D$, value restriction $\forall R.C, \exists R.T$
 - subsumption in P^{TIME} / no inconsistency
 - structural subsumption algorithm
 - e.g.: \mathcal{ALC}
 - full negation
 - reasoning can be reduced to satisfiability and is in P^{SPACE}
 - tableau algorithms
 - OWL is even more expressive, e.g. \mathcal{SHIQ} = \mathcal{ALC} +transitive and inverse \mathcal{I} roles+ \mathcal{N} and role hierarchies.

Modular Definition of Description Logics

Constructor	Syntax	Semantics
concept name	C	$C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
univ. quant.	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}. (R^{\mathcal{I}} d_1 d_2 \rightarrow d_2 \in C^{\mathcal{I}})\}$
top	\top	$\Delta^{\mathcal{I}}$
negation (\mathcal{C})	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
disjunction (\mathcal{U})	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
exist. quant. (\mathcal{E})	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}. (R^{\mathcal{I}} d_1 d_2 \wedge d_2 \in C^{\mathcal{I}})\}$
number restr. (\mathcal{N})	$\geq nR$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}} d_1 d_2\} \geq n\}$
one-of (\mathcal{O})	$\leq nR$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}} d_1 d_2\} \leq n\}$
filler (\mathcal{B})	$\{a_1, \dots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}} \text{ for some } a_i\}$
role name	R	$R^{\mathcal{I}}$
role conj. (\mathcal{R})	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
inverse roles (\mathcal{I})	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$

What we do Today

- ▶ The SW and OWL
- ▶ **Introducing Description Logics**
 - Historical background
 - Concept languages
 - **Ontologies**
- ▶ Issues in DL
- ▶ Reasoning
 - Structural subsumption
 - Tableau calculi
- ▶ Implementation and Optimisation issues

Representing Knowledge in DL

Let C_1, C_2 and C be Description Logic concepts, which are usually interpreted as sets.

A *knowledge base* Σ is a pair $\Sigma = (\mathcal{T}, \mathcal{A})$ such that

- ▶ **Terminology:** \mathcal{T} is the T(erminological)-Box, a finite set of expressions of the form $C_1 \sqsubseteq C_2$. Formulas in \mathcal{T} are called *terminological axioms*.
- ▶ **World Knowledge:** \mathcal{A} is the A(ssertional)-Box, a finite set of expressions of the forms $a : C$ or $(a, b) : R$. Formulas in \mathcal{A} are called *assertions*.

Formal Semantics and Inferences

- ▶ **Semantics:**
 - Given a universe Δ of individuals
 - Concepts are interpreted as sets over Δ ,
 - roles as binary relations
- ▶ **Models of a Knowledge Base:** Let \mathcal{I} be an interpretation and φ a terminological axiom or an assertion. Then $\mathcal{I} \models \varphi$ if
 - $\varphi = C_1 \sqsubseteq C_2$ and $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, or
 - $\varphi = a : C$ and $a^{\mathcal{I}} \in C^{\mathcal{I}}$, or
 - $\varphi = (a, b) : R$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Terminological Reasoning

- $\mathcal{I} = \{ \text{Painting} \sqsubseteq \text{Artwork} \sqcap \neg \text{Sculpture},$
 $\text{Painter} \sqsubseteq \exists \text{creates. Paintings},$
 $\text{Sculpturer} \sqsubseteq \exists \text{creates. Artwork} \sqcap \forall \text{creates. Sculpture} \}$
- ▶ **Concept satisfiability,** $\Sigma \models C \neq \perp$.
 - Check whether there is an interpretation \mathcal{I} such that $\mathcal{I} \models \Sigma$ and $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
 - **Concept unsatisfiability:** $\Sigma \models \text{Painter} \sqcap \text{Sculpturer} = \perp$.
 - ▶ **Subsumption,** $\Sigma \models C_1 \sqsubseteq C_2$.
 - Check whether for all interpretations \mathcal{I} such that $\mathcal{I} \models \Sigma$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
 - **Subsumption:** $\Sigma \models \text{Painter} \sqsubseteq \neg \text{Sculpturer}$

Assertional reasoning

- $\mathcal{A} = \{ \text{rembrandt:Artist, nightwatch:Painting,} \}$ and $\Sigma = \langle \mathcal{I}, \mathcal{A} \rangle$
 $(\text{rembrandt, nightwatch}: \text{created})$
- ▶ **Consistency,** $\Sigma \not\models \perp$.
 - Check whether there exists \mathcal{I} such that $\mathcal{I} \models \Sigma$.
 - $\Sigma \models \mathcal{A} \neq \perp$ but $\Sigma \not\models \mathcal{A} \cup \{ \text{rembrandt: Sculpturer} \} = \perp$
 - ▶ **Instance Checking,** $\Sigma \models a : C$.
 - Check whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all interpretations $\mathcal{I} \models \Sigma$.
 - $\text{rembrandt} \in \Sigma \text{Painter}$.
- Defined reasoning tasks:
- **Retrieval:** $\text{retrieve}(\text{Artists}) = \{ \text{rembrandt} \}$.
 - **Realization:** find most specific concepts in \mathcal{I} for instances in \mathcal{A}
 $\text{realize}(\text{rembrandt}) = \text{Painter}$



More on TBoxes

- ▶ **Definitorial TBoxes**
 - Defining an atomic concept A with a complex concept C : $A \sqsubseteq C$.
 - Non-cyclic definitions allow for unfolding: $\text{unfold}(C)$
 - relatively nice computational properties (although exponential blow-up possible)
 - $\Sigma \models (C = \perp)$ iff $\langle \emptyset, \emptyset \rangle \models \text{unfold}(C) = \perp$
 - $\Sigma \models (C \sqsubseteq D)$ iff $\langle \emptyset, \emptyset \rangle \models \text{unfold}(C) \sqsubseteq \text{unfold}(D)$
- ▶ **General TBoxes**
 - $C \sqsubseteq D$ for arbitrary concepts C and D
 - computationally difficult
- ▶ **Classification and Coherence** are main tasks

What we do Today

- ▶ **The SW and OWL**
- ▶ **Introducing Description Logics**
 - Historical background
 - Concept languages
 - Ontologies
 - ▶ **Issues in DL**
- ▶ **Reasoning**
 - Structural subsumption
 - Tableau calculi
- ▶ **Implementation and Optimisation issues**

Open-world assumption

- ▶ The semantics we defined is based on an open-world assumption. It follows that:

- $\mathcal{T} = \{Painter \sqsubseteq Artist \sqcap \exists creates.Painting\}$ does not imply that a painter does not create sculptures.

- $\mathcal{A} = \{rembrandt:Painter\}$ does not imply that Rembrandt did



not create sculptures.

- $\mathcal{A} = \{painted(rembrandt, nightwatch)\}$ does not imply that VanGogh did not paint the Nightwatch, and that Rembrandt did not paint the Mona Lisa.



Open World Assumption

GWtp: Johan hates the painter Vermeer, and the girl. Vermeer loves the girl, and the girl loves Maarten, who is a butcher.

Does Johan hate a painter, who is in love with a non-painter.

▶ Database:

hates	johan	vermeer
hates	johan	girl
loves	vermeer	girl
loves	girl	maarten

here we've got a closed world assumption, one single model; reasoning = model checking and the answer is **no**

▶ ABox of a Knowledge base:

$(johan, vermeer):hates$
 $(johan, girl):hates$
 $(vermeer, girl):loves$
 $(girl, maarten):loves$
 $vermeer:Painter$
 $maarten:\neg Painter$

now we've got an open-world assumption, i.e. classes of models

▶ Instance ? $johan \sqsubseteq \exists hates (Painter \sqcap \exists loves. \neg Painter)$

▶ The answer is again: "yes", but now we need case analysis

DL versus OWL-full

- ▶ Yesterday, we mentioned things that cannot be modeled in OWL DL, but in OWL full. Here are two examples:
- ▶ $\{leolion, Lion \sqsubseteq Mammal, Mammal:Species\}$
- ▶ $\{narrowerThan \text{ subproperty subclass}\}$
- ▶ $\{hasID:functionalProperty\}$

Pause

What we do Today

- ▶ The SW and OWL
- ▶ Introducing Description Logics
 - Historical background
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
- Structural subsumption
- Tableau calculi
- ▶ Implementation and Optimisation issues

Structural Subsumption for \mathcal{FL}_0

- ▶ Language has only conjunction and value restriction
- ▶ Normal form: $A_1 \sqcap \dots \sqcap A_n \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_m.C_m$
- ▶ every \mathcal{FL}_0 concept can be brought into Normal Form, as $(\forall R.(C \sqcap D) \equiv \forall R.C \sqcap \forall R.D)$.
 - ▶ Let
 - $C := A_1 \sqcap \dots \sqcap A_n \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_m.C_m$
 - $D := B_1 \sqcap \dots \sqcap B_k \sqcap \forall S_1.D_1 \sqcap \dots \sqcap \forall S_l.D_l$
 - ▶ then $C \sqsubseteq D$ iff
 1. for all B_i there exists an A_j such that $B_i = A_j$
 2. for all $\forall S_i.D_i$ there exists an $\forall R_j.C_j$ such that $S_i = R_j$ and $C_j \sqsubseteq D_i$.

What we do Today

- ▶ The SW and OWL
- ▶ Introducing Description Logics
 - Historical background
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
- Structural subsumption
- Tableau calculi
- ▶ Implementation and Optimisation issues

Subsumption and Satisfiability in \mathcal{ALC}

- ▶ Subsumption is transformed into ABox satisfiability as $\Sigma \models (C \sqsubseteq D)$ iff $\Sigma \models (C \sqcap \neg D = \perp)$ (requires negation)
- ▶ A Tableau algorithm is used to test satisfiability.
 - Try to build a model of a concept C
 - The model is represented by a set of ABoxes \mathcal{S} (a tree)
 - Nodes in \mathcal{T} correspond to individuals in the model.
 - Nodes are labeled with sets of subconcepts of C .
 - Edges are labeled with role names in C .
 - Start from the root node labeled $\{C\}$.
 - Apply expansion rules to node labels until
 - Expansion completed (the tree represents a valid model)
 - Contradictions prove there is no model
 - Many rules uses non-deterministic expansions \Rightarrow

An Example

- ▶ Let us assume that the TBox is empty
- ▶ Check whether $\exists R.A \sqcap \exists R.B$ is subsumed by $\exists R.(A \sqcap B)$
 - Check $C := \exists R.A \sqcap \exists R.B \sqcap \neg(\exists R.(A \sqcap B))$ for satisfiability
 - NNF: $C_0 := \exists R.A \sqcap \exists R.B \sqcap \forall R.(\neg A \sqcup \neg B)$ by quantifier rule and de Morgan
 - Construct an interpretation \mathcal{I} such that $C_0^{\mathcal{I}} \neq \emptyset$
 - Generate b such that $b \in (\exists R.A)^{\mathcal{I}}$ $b \in (\exists R.B)^{\mathcal{I}}$ and $b \in (\forall R.(\neg A \sqcup \neg B))^{\mathcal{I}}$
 - Introduce witnesses for existential restrictions: c and d such that $c \in A^{\mathcal{I}}$ and $d \in B^{\mathcal{I}}$
 - Impose new constraints for universal restrictions: $c : \neg A \sqcup \neg B$, $d : \neg A \sqcup \neg B$.

- Try possibilities for disjunctions: $c : \neg A$ leads to a contradiction, backtracking gives $c : \neg B$ and $d : \neg A$.
- We have a nondeterministic choice.
- This gives us a model \mathcal{I} , C is satisfiable, and the subsumption does not hold.
- ▶ What are the best data structures for these constraints such as b belongs to $(\exists R.A)^{\mathcal{I}}$

ABox Tableaux for Concept Satisfiability

- ▶ We use ABoxes (originally constraint systems) as data structures for concept satisfiability $(s:C \mid (s,t):R \mid s \neq t)$
- ▶ To show satisfiability of a concept C
 - Transform C into NNF $\rightarrow C_0$
 - Take an ABox $\{a : C_0\}$ for a new individual a
 - Apply tableau rules as long as possible
 - An ABox with an obvious contradiction (clash) is *inconsistent* otherwise it is *consistent*
 - Disjunctions and at-most rules are non-deterministic, so we get finite set \mathcal{S} of ABoxes
 - \mathcal{S} is consistent, if there is a consistent ABox in \mathcal{S}

Tableaux Rules I

- ▶ $\mathcal{A}' \rightarrow \neg \{s:C_1, s:C_2\} \cup \mathcal{A}$,
if $s:C_1 \sqcap C_2 \in \mathcal{A}$ and $\{s:C_1, s:C_2\} \not\subseteq \mathcal{A}$
- ▶ $\mathcal{A}' \rightarrow \sqcup \{s:D\} \cup \mathcal{A}$
if $s:C_1 \sqcup C_2 \in \mathcal{A}$, $\{s:C_1, s:C_2\} \cap \mathcal{A} = \emptyset$,
and $D = C_1$ or $D = C_2$;
- ▶ $\mathcal{A}' \rightarrow \exists \{(s,y):R, y:C\} \cup \mathcal{A}$
if $s:\exists R.C \in \mathcal{A}$, there is no t such that t
is a direct R -successor of s in \mathcal{A}
and $t:C$ is in \mathcal{A} , and y is a new variable;
- ▶ $\mathcal{A}' \rightarrow \forall \{t:C\} \cup \mathcal{A}$
if $s:\forall R.C$ is in \mathcal{A} , t is a filler of R for s ,
and $t:C$ is not in \mathcal{A} ;

An example tableau

To show that $\forall \text{creates.Painting} \sqcap \exists \text{creates.Painting} \sqsubseteq \exists \text{creates.Painting}$, we show that the tableau:
 $\{i : \forall \text{creates.Painting} \sqcap \exists \text{creates.Painting} \sqcap \forall \text{creates.Painting}\}$ closes.
 with \sqcap -rule, this is
 $\{i : \forall \text{creates.Painting}$
 $i : \exists \text{creates.Painting}$
 $i : \forall \text{creates.Painting}\}$
 with \exists -rule, we add
 $j:\top$
 $j:\text{Painting}$
 $j:\neg \text{Painting}$
 and this gives a clash!!!

Termination, Soundness and Completeness

- Lemma:** Let C be an \mathcal{ALC} concept and \mathcal{S} be a set of ABoxes obtained by applying the tableau rules to C . Then for every ABox $\mathcal{A} \in \mathcal{S}$:
1. **Termination:** The rule application terminates,
 2. **Soundness:** If C is satisfiable, there is a consistent ABox $\mathcal{A} \in \mathcal{S}$
 2.a) If \mathcal{A} is consistent then \rightarrow yields a consistent \mathcal{A}' ,
 2.b) If \mathcal{A} contains a clash, then \mathcal{A} has no model, and
 3. **Completeness:** If no more rules apply to \mathcal{A} , then \mathcal{A} defines a (canonical) model for C .
- Corollary:** The tableau algorithm is a PSPACE decision procedure for consistency (and subsumption) of \mathcal{ALC} concepts. And \mathcal{ALC} has the tree property.

Termination and Soundness

- Proof of Lemma:**
1. (Termination) The algorithm “increasingly” constructs a tree whose **depth** is linear in $|C|$: quantifier depth decreases from node to successor.
breadth is linear in $|C|$ (even if number in NRs are coded in binary)
 - 2.a Easy to prove (by definition of the semantics) that if \mathcal{I} is a model of \mathcal{S} and \rightarrow can be applied to \mathcal{S} to obtain \mathcal{S}' , then \mathcal{I}' is a model of \mathcal{S}' .
 - 2.b Obvious: An ABox \mathcal{A} with a clash has no model – recall definition of a clash
 $\{x : \mathcal{A}, x : \neg \mathcal{A}\} \subseteq \mathcal{A}$

Completeness

3. (Canonical model) A complete and clash-free ABox \mathcal{A} has a model.
 We construct a canonical interpretation \mathcal{I} induced by \mathcal{A} :
 the domain $\Delta^{\mathcal{I}}$ consists of the all the individual names in \mathcal{A}
 for all atomic concepts we define $\mathcal{A}^{\mathcal{I}} := \{x \mid x : \mathcal{A} \in \mathcal{A}\}$
 for all atomic roles we define $\mathcal{R}^{\mathcal{I}} := \{(x,y) \mid R(x,y) \in \mathcal{A}\}$
- ▶ Check that $x : C \in \mathcal{A}$ implies $x \in C^{\mathcal{I}}$
- ▶ \rightsquigarrow (finite) canonical tree model for input concept.

PSPACE

- ▶ To make the tableau algorithm run in PSPACE, start by recalling PSPACE = NPSPACE
- ▶ Observe that branches (ABoxes) are independent from each other
- ▶ Observe that each individual name requires linear space only
- ▶ recall that path are of length $\leq |C|$
 \rightsquigarrow each path can be stored in $\mathcal{O}(|C|^2)$
- ▶ construct/search the tree **depth first**
- ▶ re-use space from already constructed branches.

DLs with More Language Constructs

This tableau algorithm can be modified to a PSPACE decision procedure for

- ▶ **ALCC** with number restrictions ($\geq nR\ C$) and ($\leq nR\ C$)
- ▶ **ALCC** with inverse roles (e.g. `has-child-`)
 - new rule: $\mathcal{A} \rightarrow_{-1} \{(t, s) : R\} \cup \mathcal{A}$
 if $(s, t) : R^{-1}$ is in \mathcal{A} and $(t, s) : R$ is not in \mathcal{A} ;
- ▶ **ALCC** with role conjunction $\exists(R \sqcap S).C$ and $\forall(R \sqcap S).C$.
 - new rule: $\mathcal{A} \rightarrow_{\mathcal{R}} \{(s, t) : R_1, (s, t) : R_2\} \cup \mathcal{A}$
 if $(s, t) : R_1 \sqcap R_2$ is in \mathcal{A} , and
 either $(s, t) : R_1$ or $(s, t) : R_2$ is not in \mathcal{A} ;
- ▶ All this is for **concept satisfiability** with empty TBoxes

Reasoning with ontologies

- ▶ ABoxes
 - Add $x \neq y$ for all x and y occurring in \mathcal{A} getting \mathcal{A}_0 .
 - Apply rules on \mathcal{A}_0
 - Precompletion needed for termination
 - Completeness and Soundness as before
- ▶ TBox
 - TBoxes with acyclic concept definitions $\mathcal{A} \doteq C$:
 - **unfolding** (macro expansion) is easy, but suboptimal: may yield exponential blow-up
 - **lazy unfolding** (unfolding on demand) is optimal, consistency is PSPACE decidable

General TBoxes

- Extend the calculus with assertions $\forall x.x : C$
- A knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ in a language with the \mathcal{C} and \mathcal{U} constructors can be easily translated into an extended ABox \mathcal{A}_Σ by taking

$$\mathcal{A}_\Sigma = \mathcal{A} \cup \{ \forall x.x : \neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T} \}.$$

- And given a knowledge base Σ , and an assertion $a : C$

$$\Sigma \models a \in \Sigma\ C \text{ iff } \mathcal{A}_\Sigma \cup \{ a : \neg C \} \text{ is unsatisfiable.}$$

- $\mathcal{A} \rightarrow_{\forall x} \{ s : C \} \cup \mathcal{A}$
 if $\forall x.x : C$ is in \mathcal{A} , s appears in \mathcal{A} , and $s : C$ is not in \mathcal{A} .

What does that mean, what can happen?

- ▶ TBoxes with general axioms $C \sqsubseteq D$:
 - each node must be labelled with $\neg C \sqcup D$
 - quantifier depth no longer decreases
 - \rightsquigarrow termination not guaranteed
 - E.g. if $human \sqsubseteq \exists hasmother.human \in \mathcal{T}$, then $\neg human \sqcup \exists hasmother.human$ has to be added to every node.
 - Let's see what happens in an ABox $\mathcal{A} = \{ w : human \}$

Blocking

- ▶ When creating a new node, check ancestors for equal (superset) label
- ▶ If such a node is found, new node is **blocked**
- ▶ i.e. the witness for the *hasmother* relation is not created

What we do Today

- ▶ The SW and OWL
- ▶ Introducing Description Logics
 - Historical background
 - Concept languages
 - Ontologies
- ▶ Issues in DL
- ▶ Reasoning
 - Structural subsumption
 - Tableau calculi
- ▶ **Implementation and Optimisation issues**

Implementing DL Systems

Last week we saw how to implement ABox tableaux as constraint systems. If we implement these constraint systems naively we will soon run into problems

- ▶ Space usage
 - The storage required for the tableaux data-structures can grow
 - But this is rarely a serious problem in practice.
- ▶ Time usage
 - The non-deterministic expansion can overcome the prover.
 - This is a *serious* problem in practice.
 - Which is mitigated by:
 - Careful choice of algorithms.
 - Highly optimized implementations

Highly Optimized Implementations

Optimizations can be performed at two levels

- ▶ When computing the *classification* (partial ordering) of concepts.
 - Objective is to minimize number of subsumption tests
 - Can use standard order-theoretic techniques
 - E.g., use *enhanced traversal* that exploits information from previous tests.
 - Also use structural information from the KB
 - E.g., when selecting the order in which to classify concepts.
- ▶ Computing *subsumption* between concepts
 - Objective is to minimize cost of single subsumption tests
 - Small number of hard tests can dominate classification time
 - Latest DL research has mainly addressed this problem.

Optimizing Subsumption Testing

Optimizations techniques broadly fall into two categories

- ▶ Pre-processing optimizations
 - Aim is to simplify KB and facilitate subsumption testing
 - Largely algorithm independent
 - Particularly important when KB contains GCI axioms (general concept inclusion axioms)
- ▶ Algorithmic optimizations
 - Main aim is to reduce search space due to non-determinism
 - Integral part of implementations
 - But often generally applicable to search based algorithms

Pre-Processing Optimizations

Useful techniques include

- ▶ Normalization and simplification of concepts
 - Refinement of technique first used in the *KRZS* system
 - Lexically normalize and simplify all concepts in KB
 - Combine with lazy unfolding in tableaux algorithm
 - Facilitates early detection of inconsistencies (clashes)
- ▶ Absorption (simplification) of general axioms
 - Eliminate GCIs by absorbing into "definition" axioms
 - Definition axioms efficiently dealt with by lazy expansion
- ▶ Avoidance of potentially costly reasoning whenever possible
 - Normalization can discover "obvious" (un) satisfiability
 - Structural analysis can discover "obvious" subsumption

Algorithmic Optimizations

Useful techniques include

- ▶ Avoiding redundancy in search branches
 - Davis-Putnam style semantic branching search
 - Syntactic branching with no-good list
- ▶ Dependency directed backtracking
 - Backjumping
- ▶ Caching
 - Cache partial models
 - Cache satisfiability status (of labels)

Normalization and Simplification

- ▶ Normalize concepts to standard form, e.g.:
 - $\exists R.C \rightarrow \neg \forall R. \neg C$.
 - $C \sqcup D \rightarrow \neg(\neg C \sqcap \neg D)$.
- ▶ Simplify concepts, e.g.:
 - $(D \sqcap C) \sqcap (A \sqcap D) \rightarrow A \sqcap C \sqcap D$
 - $\forall R.T \rightarrow T$
 - $\dots \sqcap C \sqcap \dots \sqcap \neg C \sqcap \dots \rightarrow \perp$
- ▶ Lazily unfold concepts in tableaux algorithm
 - Use name/pointers to refer to complex concepts
 - Only add structure as required by the progress of the algorithm
 - Detect clashes between lexically equivalent concepts

Absorption I

- ▶ Reasoning w.r.t. sets of GCI axioms can be very costly
- $GCI \ C \sqsubseteq D$ adds $D \sqcup \neg C$ to every node label
- Expansion of disjunctions leads to search
- With 10 axioms and 10 nodes, search space is already 2^{100}
- GALEN or DICE contain **thousands** of axioms.
- ▶ Reasoning w.r.t. primitive definition axioms is relatively efficient
 - For $CN \sqsubseteq D$, add D **only** to node labels containing CN
 - for $CN \supseteq D$, add $\neg D$ **only** to node labels containing $\neg CN$
 - We can expand definitions lazily
 - Only add definitions **after** other local (propositional) expansions.
 - Only add definitions one step at a time.

Absorption II

- ▶ Transform GCIs into primitive definitions, e.g.:
 - $CN \sqcap C \sqsubseteq D \rightarrow CN \sqsubseteq D \sqcup \neg C$
 - $CN \sqcup C \supseteq D \rightarrow CN \supseteq D \sqcap \neg C$
- ▶ Absorb into existing primitive definitions, e.g.:
 - $CN \sqsubseteq A, CN \sqsubseteq D \sqcup \neg C \rightarrow CN \sqsubseteq A \sqcap (D \sqcup \neg C)$
 - $CN \supseteq A, CN \supseteq D \sqcap \neg C \rightarrow CN \supseteq A \sqcup (D \sqcap \neg C)$
- ▶ Use lazy expansion techniques with primitive definitions
 - Disjunctions only added to "relevant" node labels
- ▶ Performance improvements often exceptional
 - At least **four orders of magnitude** with GALEN

Dependency Directed Backtracking

- ▶ Allows rapid recovery from bad branching choices
- ▶ Most commonly used technique is **backjumping**
 - Tag concepts introduced at branch points (e.g., when expanding disjunctions)
 - Expansion rules combine and propagate tags
 - On discovering a clash, identify most recently introduced concepts involved
 - Jump back to relevant branch points **without exploring** alternative branches
- Effect is to prune away part of the search space
- Performance improvements again very good.

Caching

- ▶ Cache the satisfiability status of a node label
- Identical node labels often recur during expansion
- Avoid re-solving problems by caching satisfiability status
 - When putting x : C into \mathcal{A} , look in cache
 - use result, or add status once it has been computed
- Can use sub/super set caching to deal with similar labels
- Care required when used with blocking or inverse roles
- Significant performance gains with some kinds of problem
- ▶ Cache (partial) models of concepts
 - Use to detect "obvious" non-subsumption
 - $C \not\sqsubseteq D$ if $C \sqcap \neg D$ is satisfiable
 - $C \sqcap \neg D$ satisfiable if models of C and $\neg D$ can be merged.
 - If not, continue with standard subsumption test.

Wrap-up

- ▶ What have we seen today?
 - Description Logics as formal background to Semantic Web reasoning
 - Lots of core issues in DL research
 - History
 - Theory
 - Reasoning
 - Implementation
- ▶ What is going to come tomorrow?
 - Heiner!