Logical Foundations for the Semantic Web

Ian Horrocks and Ulrike Sattler
University of Manchester
Manchester, UK
{horrocks|sattler}@cs.man.ac.uk
Introduction
History of the Semantic Web

- Web was “invented” by Tim Berners-Lee (amongst others), a physicist working at CERN
- TBL’s original vision of the Web was much more ambitious than the reality of the existing (syntactic) Web:

  “... a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the machine-readable information space gave an accurate representation of the state of people's thoughts, interactions, and work patterns, then machine analysis could become a very powerful management tool, seeing patterns in our work and facilitating our working together through the typical problems which beset the management of large organizations.”

- TBL (and others) have since been working towards realising this vision, which has become known as the Semantic Web
  - E.g., article in May 2001 issue of Scientific American...
Realising the complete “vision” is too hard for now (probably)
But we can make a start by adding semantic annotation to web resources
Where we are Today: the Syntactic Web

[Hendler & Miller 02]
The Syntactic Web is...

- **A hypermedia, a digital library**
  - A library of documents called (web pages) interconnected by a hypermedia of links

- **A database, an application platform**
  - A common portal to applications accessible through web pages, and presenting their results as web pages

- **A platform for multimedia**
  - BBC Radio 4 anywhere in the world! Terminator 3 trailers!

- **A naming scheme**
  - Unique identity for those documents

A place where computers do the presentation (easy) and people do the linking and interpreting (hard).

Why not get computers to do more of the hard work?

[Goble 03]
Hard Work using the Syntactic Web...

Find images of Peter Patel-Schneider, Frank van Harmelen and Alan Rector...

Hard Work using the Syntactic Web…

To bee or not to bee

Search engines may be remarkable resources. Will a new 'semantic' web be clever enough to tell a flying insect from a work of music?

18 June 2003

Web searches have always been a bit hit and miss. Even when your searches are clearly defined, you'll turn up irrelevant web pages that happen to have the same keywords. Looking for details of bumble bees' flight? Google's first result points to the composer Rimsky-Korsakov…
Impossible (?) using the Syntactic Web...

- **Complex queries involving background knowledge**
  - Find information about “animals that use sonar but are not either bats or dolphins”, e.g., Barn Owl

- **Locating information in data repositories**
  - Travel enquiries
  - Prices of goods
  - Results of human genome experiments

- **Finding and using “web services”**
  - Visualise surface interactions between two proteins

- **Delegating complex tasks to web “agents”**
  - Book me a holiday next weekend somewhere warm, not too far away, and where the people speak French or English
What is the Problem?

- Consider a typical web page:
  - Markup consists of:
    - rendering information (e.g., font size and colour)
    - Hyper-links to related content
  - Semantic content is accessible to humans but not (easily) to computers...
What information can we see...

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact
Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire
Register now
On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event ...

Speakers confirmed
Tim berners-lee
Tim is the well known inventor of the Web, ...
Ian Foster
Ian is the pioneer of the Grid, the next generation internet ...
What information can a machine see...
Solution: XML markup with “meaningful” tags?

```xml
<name>

<slogan>

<participants>

<introduction>

<speaker>

<bio>
```
But What About…

<conf><image><image>

</conf>

<place><image><image>

</place>

<date><image><image>

</date>

<slogan><image><image>

</slogan>

<participants><image><image>

</participants>

<introduction><image><image>

</introduction>

<speaker><image><image>

</speaker>

<bio><image><image>

</bio>
Need to Add “Semantics”

- **External agreement** on meaning of annotations
  - E.g., Dublin Core
    - Agree on the meaning of a set of annotation tags
  - Problems with this approach
    - Inflexible
    - Limited number of things can be expressed

- **Use Ontologies to specify meaning of annotations**
  - Ontologies provide a vocabulary of terms
  - New terms can be formed by combining existing ones
  - Meaning (**semantics**) of such terms is formally specified
  - Can also specify relationships between terms in multiple ontologies
Ontology: Origins and History

Ontology in Philosophy

a philosophical discipline—a branch of philosophy that deals with the nature and the organisation of reality

- Science of Being (Aristotle, Metaphysics, IV, 1)
- Tries to answer the questions:

  What characterizes being?

  Eventually, what is being?
Ontology in Linguistics

Concept

activates

Form

Stands for

Referent

"Tank"

[Ogden, Richards, 1923]
Ontology in Computer Science

• An ontology is an engineering artifact:
  – It is constituted by a specific vocabulary used to describe a certain reality, plus
  – a set of explicit assumptions regarding the intended meaning of the vocabulary.

• Thus, an ontology describes a formal specification of a certain domain:
  – Shared understanding of a domain of interest
  – Formal and machine manipulable model of a domain of interest

“An explicit specification of a conceptualisation”
[Gruber93]
Structure of an Ontology

Ontologies typically have two distinct components:

• Names for important concepts in the domain
  – Elephant is a concept whose members are a kind of animal
  – Herbivore is a concept whose members are exactly those animals who eat only plants or parts of plants
  – Adult_Elephant is a concept whose members are exactly those elephants whose age is greater than 20 years

• Background knowledge/constraints on the domain
  – Adult_Elephants weigh at least 2,000 kg
  – All Elephants are either African_Elephants or Indian_Elephants
  – No individual can be both a Herbivore and a Carnivore
Example Ontology

The image shows a screenshot of an ontology editor. The editor is displaying a list of classes and their relationships. Some class names visible in the image include "adult_elephant", "african_animal", "african_elephant", "elephant", etc. The properties and documentation for an instance of the "african_elephant" class are also visible, indicating "Elephants from Africa."
A Semantic Web — First Steps

Make web resources more accessible to automated processes

- Extend existing rendering markup with **semantic markup**
  - Metadata annotations that describe content/function of web accessible resources
- Use Ontologies to provide **vocabulary** for annotations
  - “Formal specification” is accessible to machines

- A prerequisite is a standard web ontology language
  - Need to agree common **syntax** before we can share semantics
  - Syntactic web based on **standards** such as HTTP and HTML
"The challenge of the Semantic Web is to find a representation language powerful enough to support automated reasoning but simple enough to be usable."
Ontology Design and Deployment

- Given key role of ontologies in the Semantic Web, it will be essential to provide **tools** and **services** to help users:
  - Design and maintain high quality ontologies, e.g.:
    - **Meaningful** — all named classes can have instances
    - **Correct** — captured intuitions of domain experts
    - **Minimally redundant** — no unintended synonyms
    - **Richly axiomatised** — (sufficiently) detailed descriptions
  - Store (large numbers) of **instances** of ontology classes, e.g.:
    - Annotations from web pages
  - Answer **queries** over ontology classes and instances, e.g.:
    - Find more general/specific classes
    - Retrieve annotations/pages matching a given description
  - **Integrate** and align multiple ontologies
Ontology Languages for the Semantic Web
Resources

• Course material (including slides):
  
  http://www.cs.man.ac.uk/~horrocks/ESSLLI2003/

• Description Logic Handbook
  
  http://books.cambridge.org/0521781760.htm
Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Semantic networks
    - Topic Maps (see http://www.topicmaps.org/)
    - UML
    - RDF
  - Logic based
    - Description Logics (e.g., OIL, DAML+OIL, OWL)
    - Rules (e.g., RuleML, LP/Prolog)
    - First Order Logic (e.g., KIF)
    - Conceptual graphs
    - (Syntactically) higher order logics (e.g., LBase)
    - Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  - Probabilistic/fuzzy
- Degree of formality varies widely
  - Increased formality makes languages more amenable to machine processing (e.g., automated reasoning)
Many languages use “object oriented” model based on:

- **Objects/Instances/Individuals**
  - Elements of the domain of discourse
  - Equivalent to constants in FOL
- **Types/Classes/Concepts**
  - Sets of objects sharing certain characteristics
  - Equivalent to unary predicates in FOL
- **Relations/Properties/Roles**
  - Sets of pairs (tuples) of objects
  - Equivalent to binary predicates in FOL

- **Such languages are/can be:**
  - Well understood
  - Formally specified
  - (Relatively) easy to use
  - Amenable to machine processing
Web “Schema” Languages

- Existing Web languages extended to facilitate content description
  - **XML** → XML Schema (**XMLS**)  
  - **RDF** → RDF Schema (**RDFS**)
- **XMLS** *not* an ontology language
  - Changes format of DTDs (document schemas) to be XML
  - Adds an *extensible type hierarchy*
    - Integers, Strings, etc.
    - Can define sub-types, e.g., positive integers
- **RDFS** *is* recognisable as an ontology language
  - **Classes** and **properties**
  - **Sub/super-classes** (and properties)
  - **Range** and **domain** (of properties)
RDF and RDFS

- **RDF** stands for **Resource Description Framework**
- It is a W3C candidate recommendation ([http://www.w3.org/RDF](http://www.w3.org/RDF))
- **RDF** is **graphical formalism** (+ XML syntax + semantics)
  - for representing metadata
  - for describing the semantics of information in a machine-accessible way
- **RDFS** extends **RDF** with "**schema vocabulary**", e.g.:
  - Class, Property
  - type, subClassOf, subPropertyOf
  - range, domain
The RDF Data Model

• Statements are <subject, predicate, object> triples:
  \(<Ian, hasColleague, Uli>\)
• Can be represented as a graph:

  Ian  hasColleague  Uli

• Statements describe properties of resources
• A resource is any object that can be pointed to by a URI:
  – a document, a picture, a paragraph on the Web;
  – a book in the library, a real person (?)
  – isbn://5031-4444-3333
  – ...
• Properties themselves are also resources (URIs)
URIs

• URI = Uniform Resource Identifier
• "The generic set of all names/addresses that are short strings that refer to resources"
• URLs (Uniform Resource Locators) are a particular type of URI, used for resources that can be accessed on the WWW (e.g., web pages)
• In RDF, URIs typically look like “normal” URLs, often with fragment identifiers to point at specific parts of a document:
  – http://www.somedomain.com/some/path/to/file#fragmentID
Linking Statements

- The subject of one statement can be the object of another
- Such collections of statements form a directed, labeled graph

```
Ian hasColleague Uli

Uli hasColleague Carole

Uli hasHomePage http://www.cs.mam.ac.uk/~sattler
```

- Note that the object of a triple can also be a “literal” (a string)
RDF Syntax

- RDF has an XML syntax that has a specific meaning:
- Every `Description` element describes a resource
- Every attribute or nested element inside a `Description` is a property of that Resource
- We can refer to resources by using URIs

```xml
<Description about="some.uri/person/ian_horrocks">
    <hasColleague resource="some.uri/person/uli_sattler"/>
</Description>
<Description about="some.uri/person/uli_sattler">
    <hasHomePage>http://www.cs.mam.ac.uk/~sattler</hasHomePage>
</Description>
<Description about="some.uri/person/carole_goble">
    <hasColleague resource="some.uri/person/uli_sattler"/>
</Description>
```
RDF Schema (RDFS)

- RDF gives a formalism for metadata annotation, and a way to write it down in XML, but it does not give any special meaning to vocabulary such as `subClassOf` or `type`
  - Interpretation is an arbitrary binary relation

- RDF Schema allows you to define vocabulary terms and the relations between those terms
  - It gives “extra meaning” to particular RDF predicates and resources
  - This “extra meaning”, or semantics, specifies how a term should be interpreted
RDFS Examples

• RDF Schema terms (just a few examples):
  – Class
  – Property
  – type
  – subClassOf
  – range
  – domain

• These terms are the RDF Schema building blocks (constructors) used to create vocabularies:
  <Person,type,Class>
  <hasColleague,type,Property>
  <Professor,subClassOf,Person>
  <Carole,type,Professor>
  <hasColleague,range,Person>
  <hasColleague,domain,Person>
RDF/RDFS “Liberality”

- No distinction between classes and instances (individuals)
  - `<Species, type, Class>`
  - `<Lion, type, Species>`
  - `<Leo, type, Lion>`
- Properties can themselves have properties
  - `<hasDaughter, subPropertyOf, hasChild>`
  - `<hasDaughter, type, familyProperty>`
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  - `<type, range, Class>`
  - `<Property, type, Class>`
  - `<type, subPropertyOf, subClassOf>`
RDF/RDFS Semantics

- RDF has “Non-standard” semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)
Semantics and Model Theories

- Ontology/KR languages aim to model (part of) world
- Terms in language correspond to entities in world
- Meaning given by, e.g.:
  - Mapping to another formalism, such as FOL, with own well defined semantics
  - or a bespoke Model Theory (MT)
- **MT defines relationship between syntax and interpretations**
  - Can be many interpretations (models) of one piece of syntax
  - Models supposed to be analogue of (part of) world
    - E.g., elements of model correspond to objects in world
  - Formal relationship between syntax and models
    - Structure of models reflect relationships specified in syntax
  - Inference (e.g., subsumption) defined in terms of MT
    - E.g., $\mathcal{T} \models A \sqsubseteq B$ iff in every model of $\mathcal{T}$, $\text{ext}(A) \subseteq \text{ext}(B)$
• Ontology/KR languages aim to model (part of) world
• Terms in language correspond to entities in world
• Meaning given by, e.g.:
  – Mapping to another formalism, such as FOL, with own well defined semantics
  – or a bespoke Model Theory (MT)
• MT defines relationship between syntax and interpretations
  – Can be many interpretations (models) of one piece of syntax
  – Models supposed to be analogue of (part of) world
    • E.g., elements of model correspond to objects in world
  – Formal relationship between syntax and models
    • Structure of models reflect relationships specified in syntax
  – Inference (e.g., subsumption) defined in terms of MT
    • E.g., $\mathcal{T} \models A \sqsubseteq B$ iff in every model of $\mathcal{T}$, $\text{ext}(A) \subseteq \text{ext}(B)$
RDF/RDFS Semantics

- RDF has “Non-standard” semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)
- In RDF MT, an interpretation $I$ of a vocabulary $V$ consists of:
  - $IR$, a non-empty set of resources
  - $IS$, a mapping from $V$ into $IR$
  - $IP$, a distinguished subset of $IR$ (the properties)
    - A vocabulary element $v \in V$ is a property iff $IS(v) \in IP$
  - $IEXT$, a mapping from $IP$ into the powerset of $IR \times IR$
    - i.e., a set of elements $<x,y>$, with $x,y$ elements of $IR$
  - $IL$, a mapping from typed literals into $IR$
- Class interpretation $ICEXT$ simply induced by $IEXT(IS(type))$
  - $ICEXT(C) = \{ x | <x,C> \in IEXT(IS(type)) \}$
Example RDF/RDFS Interpretation
RDFS Interpretations

• RDFS adds extra constraints on interpretations
  – E.g., interpretations of \(<C, \text{subClassOf}, D>\) constrained to those where ICEXT(IS(C)) ⊆ ICEXT(IS(D))

• Can deal with triples such as
  – \(<\text{Species}, \text{type}, \text{Class}>\)
    \(<\text{Lion}, \text{type}, \text{Species}>\)
    \(<\text{Leo}, \text{type}, \text{Lion}>\)
  – \(<\text{SelfInst}, \text{type}, \text{SelfInst}>\)

• And even with triples such as
  – \(<\text{type}, \text{subPropertyOf}, \text{subClassOf}>\)

• But not clear if meaning matches intuition (if there is one)
Problems with RDFS

• RDFS too weak to describe resources in sufficient detail
  – No localised range and domain constraints
    • Can’t say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  – No existence/cardinality constraints
    • Can’t say that all instances of person have a mother that is also a person, or that persons have exactly 2 parents
  – No transitive, inverse or symmetrical properties
    • Can’t say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  – ...

• Difficult to provide reasoning support
  – No “native” reasoners for non-standard semantics
  – May be possible to reason via FO axiomatisation
Web Ontology Language Requirements

Desirable features identified for Web Ontology Language:

- Extends existing Web standards
  - Such as XML, RDF, RDFS
- Easy to understand and use
  - Should be based on familiar KR idioms
- Formally specified
- Of “adequate” expressive power
- Possible to provide automated reasoning support
From RDF to OWL

- Two languages developed to satisfy above requirements
  - OIL: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - DAML-ONT: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce DAML+OIL
  - Development was carried out by “Joint EU/US Committee on Agent Markup Languages”
  - Extends (“DL subset” of) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed
  - WebOnt group developed OWL language based on DAML+OIL
  - OWL language now a W3C Candidate Recommendation
  - Will soon become Proposed Recommendation
OWL Language

• Three species of OWL
  – **OWL full** is union of OWL syntax and RDF
  – **OWL DL** restricted to FOL fragment (≈ DAML+OIL)
  – **OWL Lite** is “easier to implement” subset of OWL DL

• Semantic layering
  – **OWL DL** ≈ **OWL full** within DL fragment
  – DL semantics officially definitive

• **OWL DL** based on **SHIQ** Description Logic
  – In fact it is equivalent to **SHOIN(D_n)** DL

• **OWL DL** Benefits from many years of DL research
  – Well defined **semantics**
  – **Formal properties** well understood (complexity, decidability)
  – Known **reasoning algorithms**
  – **Implemented systems** (highly optimised)
(In)famous “Layer Cake”

• Relationship between layers is not clear
• OWL DL extends “DL subset” of RDF
# OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>Modal Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${\text{john}} \sqcup {\text{mary}}$</td>
<td>$x_1 \lor \ldots \lor x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$[P]C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\langle P\rangle C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$ hasChild</td>
<td>$[P]_{n+1}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$ hasChild</td>
<td>$\langle P\rangle_n$</td>
</tr>
</tbody>
</table>

- **XMLS datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists$ hasAge.nonNegativeInteger
- **Arbitrarily complex nesting** of constructors
  - E.g., Person $\sqcap \forall$ hasChild.Doctor $\sqcup \exists$ hasChild.Doctor
E.g., Person $\sqcap \forall$hasChild.Doctor $\sqcap \exists$hasChild.Doctor:

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
## OWL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \subseteq C_2$</td>
<td>$\text{Human} \subseteq \text{Animal} \cap \text{Biped}$</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>$\text{Man} \equiv \text{Human} \cap \text{Male}$</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \subseteq \neg C_2$</td>
<td>$\text{Male} \subseteq \neg \text{Female}$</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>${\text{President Bush}} \equiv {\text{G.W. Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \subseteq \neg {x_2}$</td>
<td>${\text{john}} \subseteq \neg {\text{peter}}$</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \subseteq P_2$</td>
<td>$\text{hasDaughter} \subseteq \text{hasChild}$</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>$\text{cost} \equiv \text{price}$</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td>$\text{hasChild} \equiv \text{hasParent}^-$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \subseteq P$</td>
<td>$\text{ancestor}^+ \subseteq \text{ancestor}$</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$\top \subseteq \leq 1P$</td>
<td>$\top \subseteq \leq 1\text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$\top \subseteq \leq 1P^-$</td>
<td>$\top \subseteq \leq 1\text{hasSSN}^-$</td>
</tr>
</tbody>
</table>

- **Axioms (mostly) reducible to inclusion (⊆)**
  - $C \equiv D$ iff both $C \subseteq D$ and $D \subseteq C$
XML Schema Datatypes in OWL

- OWL supports XML Schema primitive datatypes
  - E.g., integer, real, string, ...

- Strict separation between “object” classes and datatypes
  - Disjoint interpretation domain $\Delta_d$ for datatypes
    - For a datavalue $d$, $d^I \subseteq \Delta_d$
    - And $\Delta_d \cap \Delta^I = \emptyset$
  - Disjoint “object” and datatype properties
    - For a datatype property $P$, $P^I \subseteq \Delta^I \times \Delta_d$
    - For object property $S$ and datatype property $P$, $S^I \cap P^I = \emptyset$

- Equivalent to the “$D_n$” in $SHOIN(D_n)$
Why Separate Classes and Datatypes?

• Philosophical reasons:
  – Datatypes structured by built-in predicates
  – Not appropriate to form new datatypes using ontology language

• Practical reasons:
  – Ontology language remains simple and compact
  – Semantic integrity of ontology language not compromised
  – Implementability not compromised — can use hybrid reasoner
    • Only need sound and complete decision procedure for: $\Delta^I_1 \cap \ldots \cap \Delta^I_n$, where $\Delta$ is a (possibly negated) datatype
OWL DL Semantics

• Mapping OWL to equivalent DL ($\mathit{SHOIN} \mathcal{D}_n$):
  – Facilitates provision of reasoning services (using DL systems)
  – Provides well defined semantics

• DL semantics defined by interpretations: $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where
  – $\Delta^\mathcal{I}$ is the domain (a non-empty set)
  – $\cdot^\mathcal{I}$ is an interpretation function that maps:
    • Concept (class) name $A \rightarrow$ subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$
    • Role (property) name $R \rightarrow$ binary relation $R^\mathcal{I}$ over $\Delta^\mathcal{I}$
    • Individual name $i \rightarrow i^\mathcal{I}$ element of $\Delta^\mathcal{I}$
DL Semantics

- Interpretation function $\mathcal{I}$ extends to concept expressions in an obvious(ish) way, i.e.:

\[
\begin{align*}
(C \cap D)^\mathcal{I} & = C^\mathcal{I} \cap D^\mathcal{I} \\
(C \cup D)^\mathcal{I} & = C^\mathcal{I} \cup D^\mathcal{I} \\
(\neg C)^\mathcal{I} & = \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
\{x\}^\mathcal{I} & = \{x^\mathcal{I}\} \\
(\exists R.C)^\mathcal{I} & = \{x \mid \exists y. \langle x, y \rangle \in R^\mathcal{I} \land y \in C^\mathcal{I}\} \\
(\forall R.C)^\mathcal{I} & = \{x \mid \forall y. \langle x, y \rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\} \\
(\leq nR)^\mathcal{I} & = \{x \mid \#\{y \mid \langle x, y \rangle \in R^\mathcal{I}\} \leq n\} \\
(\geq nR)^\mathcal{I} & = \{x \mid \#\{y \mid \langle x, y \rangle \in R^\mathcal{I}\} \geq n\}
\end{align*}
\]
DL Knowledge Bases (Ontologies)

- An OWL ontology maps to a DL Knowledge Base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
  - $\mathcal{T}$ (Tbox) is a set of axioms of the form:
    - $C \subseteq D$ (concept inclusion)
    - $C \equiv D$ (concept equivalence)
    - $R \subseteq S$ (role inclusion)
    - $R \equiv S$ (role equivalence)
    - $R^+ \subseteq R$ (role transitivity)
  - $\mathcal{A}$ (Abox) is a set of axioms of the form
    - $x \in D$ (concept instantiation)
    - $\langle x, y \rangle \in R$ (role instantiation)
- Two sorts of Tbox axioms often distinguished
  - “Definitions”
    - $C \subseteq D$ or $C \equiv D$ where $C$ is a concept name
  - General Concept Inclusion axioms (GCIs)
    - $C \subseteq D$ where $C$ in an arbitrary concept
Knowledge Base Semantics

- An interpretation $\mathcal{I}$ satisfies (models) an axiom $A$ ($\mathcal{I} \models A$):
  - $\mathcal{I} \models C \subseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$
  - $\mathcal{I} \models C \equiv D$ iff $C^\mathcal{I} = D^\mathcal{I}$
  - $\mathcal{I} \models R \subseteq S$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$
  - $\mathcal{I} \models R \equiv S$ iff $R^\mathcal{I} = S^\mathcal{I}$
  - $\mathcal{I} \models R^+ \subseteq R$ iff $(R^\mathcal{I})^+ \subseteq R^\mathcal{I}$
  - $\mathcal{I} \models x \in D$ iff $x^\mathcal{I} \in D^\mathcal{I}$
  - $\mathcal{I} \models \langle x, y \rangle \in R$ iff $(x^\mathcal{I}, y^\mathcal{I}) \in R^\mathcal{I}$

- $\mathcal{I}$ satisfies a Tbox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) iff $\mathcal{I}$ satisfies every axiom $A$ in $\mathcal{T}$
- $\mathcal{I}$ satisfies an Abox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) iff $\mathcal{I}$ satisfies every axiom $A$ in $\mathcal{A}$
- $\mathcal{I}$ satisfies an KB $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) iff $\mathcal{I}$ satisfies both $\mathcal{T}$ and $\mathcal{A}$
Inference Tasks

- Knowledge is **correct** (captures intuitions)
  - $\mathcal{C}$ **subsumes** $\mathcal{D}$ w.r.t. $\mathcal{K}$ iff for **every model** $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{C}^I \subseteq \mathcal{D}^I$

- Knowledge is **minimally redundant** (no unintended synonyms)
  - $\mathcal{C}$ is **equivalent to** $\mathcal{D}$ w.r.t. $\mathcal{K}$ iff for **every model** $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{C}^I = \mathcal{D}^I$

- Knowledge is **meaningful** (classes can have instances)
  - $\mathcal{C}$ is **satisfiable** w.r.t. $\mathcal{K}$ iff there exists **some model** $\mathcal{I}$ of $\mathcal{K}$ s.t. $\mathcal{C}^I \neq \emptyset$

- **Querying** knowledge
  - $x$ is an **instance** of $\mathcal{C}$ w.r.t. $\mathcal{K}$ iff for **every model** $\mathcal{I}$ of $\mathcal{K}$, $x^I \in \mathcal{C}^I$
  - $\langle x, y \rangle$ is an **instance** of $\mathcal{R}$ w.r.t. $\mathcal{K}$ iff for, **every model** $\mathcal{I}$ of $\mathcal{K}$, $(x^I,y^I) \in \mathcal{R}^I$

- **Knowledge base consistency**
  - A KB $\mathcal{K}$ is **consistent** iff there exists **some model** $\mathcal{I}$ of $\mathcal{K}$
3. Reasoning Services and Algorithms

Help knowledge engineer and users to build and use ontologies

Ian Horrocks and Ulrike Sattler
University of Manchester
Manchester, UK
{horrocks|sattler}@cs.man.ac.uk
Plan for today

1. “useful” reasoning services
2. relationship between DLs and other logics (briefly)
3. system demonstration
4. tableau algorithm for $\mathcal{ALC}$ and how to prove its correctness
5. how to extend this algorithm to DAML+OIL and OWL
Remember: Complexity of Ontology engineering

Remember ontology engineering tasks:

- design
- evolution
- inter-operation and Integration
- deployment

Further complications are due to

- sheer size of ontologies
- number of persons involved
- users not being knowledge experts
- natural laziness
- etc.
Reasoning Services: what we might want in the Design Phase

- be warned when making **meaningless** statements
  - test **satisfiability** of defined concepts
    \[ \text{SAT}(C, T) \text{ iff there is a model } I \text{ of } T \text{ with } C^I \neq \emptyset \]
    unsatisfiable, defined concepts are signs of faulty modelling

- see **consequences** of statements made
  - test defined concepts for **subsumption**
    \[ \text{SUBS}(C, D, T) \text{ iff } C^I \subseteq D^I \text{ for all model } I \text{ of } T \]
    unwanted or missing subsumptions are signs of imprecise/faulty modelling

- see **redundancies**
  - test defined concepts for **equivalence**
    \[ \text{SUBS}(C, D, T) \text{ iff } C^I = D^I \text{ for all model } I \text{ of } T \]
    knowing about “redundant” classes helps avoid misunderstandings
Reasoning Services: what we might want when Modifying Ontologies

- the same system services as in the design phase, plus
- automatic generation of concept definitions from examples
  ➔ given individuals $o_1, \ldots, o_n$ with assertions ("ABox") for them, create
  a (most specific) concept $C$ such that each $o_i \in C^\mathcal{I}$ in each model $\mathcal{I}$ of $\mathcal{T}$
  “non-standard inferences”

- automatic generation of concept definitions for too many siblings
  ➔ given concepts $C_1, \ldots, C_n$, create
  a (most specific) concept $C$ such that $\text{SUBS}(C_i, C, \mathcal{T})$
  “non-standard inferences”

- etc.
Reasoning Services: what we might want when Integrating and Using Ontologies

For integration:

- the same system services as in the design phase, plus
- the possibility to abstract from concepts to patterns and compare patterns
  e.g., compute those concepts $D$ defined in $T_2$ such that

\[
\text{SUBS}(\text{Human} \sqcap (\forall \text{child.}(X \sqcap \forall \text{child.Y})), D, T_1 \cup T_2)
\]

“non-standard inferences”

When using ontologies:

- the same system services as in the design phase and the integration phase, plus
- automatic classification of individuals
  given individual $o$ with assertions, return all defined concepts $D$ such that

\[
o \in D^\mathcal{I} \text{ for all models } \mathcal{I} \text{ of } T
\]
(many) reasoning problems are inter-reducible:

\[
\begin{align*}
\text{EQUIV}(C, D, T) & \iff \text{sub}(C, D, T) \text{ and } \text{sub}(D, C, T) \\
\text{SUBS}(C, D, T) & \iff \text{not SAT}(C \sqcap \neg D, T) \\
\text{SAT}(C, T) & \iff \text{not SUBS}(C, A \sqcap \neg A, T) \\
\text{SAT}(C, T) & \iff \text{cons}(\{o : C\}, T)
\end{align*}
\]

⇒ In the following, we concentrate on \(\text{SAT}(C, T)\)
We know SAT is reducible to co-SUBS and vice versa.

Hence SAT is undecidable iff SUBS is.

SAT is semi-decidable iff co-SUBS is.

⇒ if SAT is undecidable but semi-decidable, then

there exists a complete SAT algorithm:

\[ \text{SAT}(C, T) \iff \text{"satisfiable"}, \text{ but might not terminate if not } \text{SAT}(C, T) \]

there is a complete co-SUBS algorithm:

\[ \text{SUBS}(C, T) \iff \text{"subsumption"}, \text{ but might not terminate if } \text{SUBS}(C, D, T) \]

1. Do expressive ontology languages exist with decidable reasoning problems?

2. Is there a practical difference between ExpTime-hard and non-terminating?
Do Reasoning Services need to be Decidable?

We know \( \text{SAT} \) is reducible to \( \text{co-SUBS} \) and vice versa.

Hence \( \text{SAT} \) is undecidable iff \( \text{SUBS} \) is undecidable.
\( \text{SAT} \) is semi-decidable iff \( \text{co-SUBS} \) is semi-decidable.

\( \rightarrow \) if \( \text{SAT} \) is undecidable but semi-decidable, then

there exists a complete \( \text{SAT} \) algorithm:
\[ \text{SAT}(C, T) \iff \text{"satisfiable"}, \text{but might not terminate if not } \text{SAT}(C, T) \]

there is a complete \( \text{co-SUBS} \) algorithm:
\[ \text{SUBS}(C, T) \iff \text{"subsumption"}, \text{but might not terminate if } \text{SUBS}(C, D, T) \]

1. Do expressive ontology languages exist with decidable reasoning problems?
   Yes: DAML+OIL and OWL

2. Is there a practical difference between ExpTime-hard and non-terminating?
   let’s see
(slide with translation)

- **SHI** is a fragment of first order logic
- **SHIQ** is a fragment of first order logic with counting quantifiers and equality
- **SHI** without transitivity is a fragment of first order with two variables
- **ALC** is a notational variant of the multi-modal logic K
  - inverse roles are closely related to converse/past modalities
  - transitive roles are closely related to transitive frames/axiom 4
  - number restrictions are closely related to deterministic programs in PDL
system demonstration
Deciding Satisfiability of $\mathit{SHIQ}$

Remember: $\mathit{SHIQ}$ is OWL-DL without datatypes and individuals

Next: tableau-based decision procedure for SAT $(C, \mathcal{T})$
we start with $\mathcal{ALC} (\sqcap, \sqcup, \neg, \exists, \forall)$ instead of $\mathit{SHIQ}$ and SAT$(C, \emptyset)$

Technical: all concepts are assumed to be in Negation Normal Form
transform $C$ into equivalent $\text{NNF}(C)$ by pushing negation inwards, using

\[
\begin{align*}
\neg(C \cap D) & \equiv \neg C \cup \neg D \\
\neg(C \cup D) & \equiv \neg C \cap \neg D \\
\neg(\exists R.C) & \equiv (\forall R.\neg C) \\
\neg(\forall R.C) & \equiv (\exists R.\neg C)
\end{align*}
\]

The algorithm decides SAT$(C, \emptyset)$ by trying to construct a model $\mathcal{I}$ for $C$
A Tableau Algorithm for $\mathcal{ALC}$

The algorithm works on a completion tree with

- nodes $x$ corresponding to elements $x \in \Delta^\mathcal{I}$
- node labels $C \in \mathcal{L}(x)$ meaning $x \in C^\mathcal{I}$
- edge labels $(x, R, y)$ representing role successorships $(x, y) \in R^\mathcal{I}$

starts with root $x$ with $\mathcal{L}(x) = \{C\}$

applies rules that infer constraints on $\mathcal{I}$

answers “$C$ is satisfiable” if rules

- can be applied (non-deterministic rules!)
- exhaustively (until no more rules apply)
- without generating a clash (node label with $\{A, \neg A\} \subseteq \mathcal{L}(x)$)

Rules: see slide
Example: $A \land \exists R.A \land \forall R.(\neg A \lor B)$ see blackboard
A Tableau Algorithm for $\mathcal{ALC}$

**Theorem** The tableau algorithm decides satisfiability of $\mathcal{ALC}$ concepts

**Lemma** let $C$ be an $\mathcal{ALC}$ concept in NNF.

(a) the $t$-algorithm terminates when started with $C$
(b) $\text{SAT}(C) \iff$ rules can be applied exhaustively without generating a clash

**Proof:** (a) the $t$-algorithm builds a completion tree
- in a monotonic way
- whose depth is bounded by $|C|$: if $y$ is an $R$-successor of $x$, then
  $$\max\{|D| \mid D \in \mathcal{L}(y)\} < \max\{|D| \mid D \in \mathcal{L}(x)\}$$
- whose breadth is bounded by $|C|$: at most one successor per $\exists R.D \in \text{sub}(C)$
Lemma let $C$ be an $\mathcal{ALC}$ concept in NNF.

(a) the t-algorithm terminates when started with $C$
(b) $\text{SAT}(C) \iff$ rules can be applied exhaustively without generating a clash

Proof: (b) $\iff$ the clash-free, complete tree built for $C$ corresponds to a model $\mathcal{I}$ of $C$:

- set $\Delta^\mathcal{I}$ to the nodes
- set $x \in A^\mathcal{I}$ iff $A \in \mathcal{L}(x)$
- set $(x, y) \in R^\mathcal{I}$ iff $(x, R, y)$ in completion tree
- prove that, if $D \in \mathcal{L}(x)$, then $x \in D^\mathcal{I}$, by induction on structure of $D$

Details: see blackboard

(this finishes the proof since $C \in \mathcal{L}(x_0)$)
Lemma let $C$ be an $\mathcal{ALC}$ concept in NNF.

(a) the t-algorithm terminates when started with $C$
(b) $\text{SAT}(C) \iff$ rules can be applied exhaustively without generating a clash

Proof: (b) $\Rightarrow$ use a model $\mathcal{I}$ of $C$ with $a \in C^\mathcal{I}$ to steer rule application via mapping

$$\pi : \text{nodes of completion tree into } \Delta^\mathcal{I}$$

built together with completion tree that satisfies

1. if $C \in \mathcal{L}(x)$, then $\pi(x) \in C^\mathcal{I}$
2. if $(x, R, y)$, then $(\pi(x), \pi(y)) \in R^\mathcal{I}$

Existence of $\pi$ implies clash-freeness of tree (1), termination is already proven

Construction of $\pi$: see blackboard with previous example
A Tableau Algorithm for \(\mathcal{ALC}\) with TBoxes

Remember:

- A GCI is of the form \(C \sqsubseteq D\) for \(C, D\) (complex) concepts
- A (general) TBox is a finite set of GCIs
- \(\mathcal{I}\) satisfies \(C \sqsubseteq D\) iff \(C^\mathcal{I} \subseteq D^\mathcal{I}\)
- \(\mathcal{I}\) is a model of TBox \(\mathcal{T}\) iff \(\mathcal{I}\) satisfies each GCI in \(\mathcal{T}\)
- recall translation of GCIs into FOL

Extend \(\mathcal{ALC}\) tableau algorithm to decide \(\text{SAT}(C, \mathcal{T})\) for TBox

\[\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leq i \leq n\}\]

Add a new rule

\[\rightarrow_{\text{GCI}}: \quad \text{If } (\neg C_i \sqcup D_i) \not\in \mathcal{L}(x) \text{ for some } 1 \leq i \leq n\]

\[\text{Then } \mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{(\neg C_i \sqcup D_i)\}\]
Example: Consider TBox \( \{ C : \exists R.C \} \). Is \( C \) satisfiable w.r.t. this TBox?
Example: Consider TBox \( \{ C \sqsubseteq \exists R.C \} \). Is \( C \) satisfiable w.r.t. this TBox?

**tableau algorithm no longer terminates!**

Reason: the size of concepts no longer decreases along paths in a completion tree

Observation: most nodes in example completion tree are similar, algorithm is repeating the same nodes

Solution: Regain termination with cycle-detection

if \( \mathcal{L}(x) \) and \( \mathcal{L}(y) \) are “very similar”, only extend \( \mathcal{L}(x) \)
A tableau algorithm for $\mathcal{ALC}$ with general TBoxes: Cycle-detection

**Blocking:**

- $x$ is **directly blocked** if it has an ancestor $y$ with $\mathcal{L}(x) \subseteq \mathcal{L}(y)$
- in this case (and if $y$ is the “closest” such node to $x$), $x$ is blocked by $y$
- A node is **blocked** if it is directly blocked or one of its ancestors is blocked

⊕ restrict the application of all rules to nodes which are not blocked

\[\sim\text{ Tableau algorithm for } \mathcal{ALC} \text{ w.r.t. TBoxes}\]

**Example:** check previous example

**Theorem** The extended t-algorithm decides satisfiability of $\mathcal{ALC}$ concepts w.r.t. TBoxes
Lemma  let $C$ be an $\mathcal{ALC}$ concept and $T$ a TBox in in NNF.

(a) the t-algorithm terminates when started with $C$ and $T$
(b) $\text{SAT}(C, T)$ \iff rules can be applied exhaustively without generating a clash

Proof: (a) the t-algorithm builds a completion tree

- in a monotonic way
- whose depth is bounded by $2^{|C|}$:
  - on any longer path, blocking would occur and
  - paths with blocked nodes do not become longer
- whose breadth is bounded by $|C|$:
  at most one successor per $\exists R.D \in \text{sub}(C)$
Lemma let $C$ be an $\mathcal{ALC}$ concept and $T$ a TBox in in NNF.

(a) the t-algorithm terminates when started with $C$ and $T$
(b) $\text{SAT}(C, T) \iff$ rules can be applied exhaustively without generating a clash

Proof: (b) $\Rightarrow$ similar to previous

$\iff$ the clash-free, complete tree built for $C$ corresponds to a model $\mathcal{I}$ of $C$ and $T$:

- set $\Delta^\mathcal{I}$ to the unblocked nodes
- set $x \in A^\mathcal{I}$ iff $A \in \mathcal{L}(x)$
- set $(x, y) \in R^\mathcal{I}$ iff $(x, R, y)$ or $(x, R, y')$ and $y$ blocks $y$
- prove that, if $D \in \mathcal{L}(x)$, then $x \in D^\mathcal{I}$, by induction on structure of $D$
  Details: see blackboard

(this finishes the proof since $C \in \mathcal{L}(x_0)$ and $\neg C_i \sqcup D_i \in \mathcal{L}(x)$, for all $i, x$)
A tableau algorithm for $\text{SHIQ}$: Transitive Roles

Remember: $\text{SHIQ}$ allows to state transitivity of roles $\text{trans}(R)$

Problem: if $\forall R.C \in \mathcal{L}(x)$ for $R$ transitiv and $(x, R, y)$ and $(y, R, z)$ in completion tree, $C$ must go to $\mathcal{L}(z)$

Solution 1: add edge $(x, R, z)$ destroys handy tree structure

Solution 2: new $\forall$ rule

$\rightarrow^+:$ If $\forall R.C \in \mathcal{L}(x)$ and $(x, R, y)$ with $R$ transitive and $\forall R.C \not\in \mathcal{L}(y)$

Then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\forall R.C\}$

Proof of “the Lemma” is similar to previous case, but for model construction:

- if $\text{trans}(R)$: $R^I = \{(x, y) \mid (x, R, y) \text{ or } (x, R, y') \text{ and } y' \text{ blocks } y\}^+$
Remember: \textit{SHIQ} allows to state role inclusions $R \sqsubseteq S$

Problem: if $(x, R, y)$ and $R \sqsubseteq^+ S$, then $(x, y) \in S^I$

Solution: define \textit{y being an $S$-successor of $x$} if $(x, R, y)$ for some $R \sqsubseteq^* S$ in rules, replace “$(x, R, y)$” with “\textit{y is $R$-successor of $x$}”

Problem2: if $\forall S.C \in \mathcal{L}(x)$ and $R$ transitive and $R \sqsubseteq S$ and $(x, R, y)$ and $(y, R, z)$ in completion tree, then $C$ must go to $\mathcal{L}(z)$

Solution: modify new $\forall$ rule

$$\rightarrow^{+}_{\forall} : \text{If } \forall S.C \in \mathcal{L}(x), \text{ } x \text{ has } R\text{-successor } y \text{ for }$$

$$R \text{ transitive and } R \sqsubseteq^* S \text{ and } \forall R.C \notin \mathcal{L}(y)$$

Then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\forall R.C\}$
A tableau algorithm for \( \text{SHIQ} \): Inverse Roles

Remember: \( \text{SHIQ} \) allows to use role names and inverse roles \( R^- \), e.g. \( \forall R^- . C \)

Problem 1: concepts need get **pushed up** the completion tree

Example: \( \exists R. (A \land \forall R^- . (B \land \exists S^- . (B \land \forall S . \neg A))) \)

Solution: treat role names and inverse roles symmetrically

define \( R \)-neighbours and replace “successor” with “neighbour” in rules

Problem 2: algorithm not correct

Example: \( \text{SAT} (A \land \forall R^- . (A \land \neg A), \ \{ A \cup \exists R . C \}) \)

Solution: modify **direct blocking condition**: \( x \) blocks \( y \) if \( \mathcal{L}(x) = \mathcal{L}(y) \)
A tableau algorithm for *SHIQ*: Number Restrictions

Remember: *SHIQ* allows to use number restrictions \((\geq nR.C), (\leq nR.C)\)

Obvious: new rules that generate \(R\)-successors \(y_i\) of \(x\) for \((\geq nr.C) \in \mathcal{L}(x)\)

new rules that identify surplus \(R\)-successors of \(x\) with \((\leq nr.C) \in \mathcal{L}(x)\)

Example: \((\geq 2R.A) \cap (\geq 2R.(A \cap B)) \cap (\leq 3S.A)\)

Less obvious: new choose rule required

Example: \((\geq 3R.A) \cap (\leq 1R.A) \cap (\leq 1R.\neg A)\)

Tricky: new blocking condition required

Proofs of Lemma become more demanding, i.e., model construction uses enhanced “unravelling” to construct possibly infinite models...
Models of \textit{SHIQ}\\

For \textit{SHIQ} without number restriction, we built finite models

ok since \textit{SHI} has finite model property, i.e.,
\[ \text{SAT}(C, T) \Rightarrow C, T \text{ have a finite model} \]

For full \textit{SHIQ}, we built infinite tree models

ok since \textit{SHIQ} has tree model property, i.e.,
\[ \text{SAT}(C, T) \Rightarrow C, T \text{ have a tree model} \]

ok since \textit{SHIQ} lacks finite model property, i.e.,
there are \( C \) and \( T \) with \( \text{SAT}(C, T) \),
but each of their models is infinite

Example: for \( F \subseteq R \) and \( R \) transitive,
\[ \neg A \land \exists F. A \land \forall R. (A \land \exists F. A \land (\leq 1 F^{-} \top)) \]
is satisfiable, but each model has an infinite \( F \)-chain (blackboard)
4. Reasoning Services and Algorithms

Help knowledge engineer and users to build and use ontologies

Ian Horrocks and Ulrike Sattler
University of Manchester
Manchester, UK
{horrocks|sattler}@cs.man.ac.uk
Plan for today

1. a few interesting complexity results for DLs
2. why full DAML+OIL and OWL-DL are more complex
3. some interesting undecidability results
4. implementing and optimising tableau algorithm
Yesterday, we have seen a tableau-based algorithm that decides

satisfiability of $SHIQ$ concepts w.r.t. $SHIQ$ TBoxes

Still missing from $SHIQ$ to OWL-DL:

- **data types** (integers, strings, with comparisons)
  
  e.g., $\text{Human} \sqcap \exists \text{age.}>18$
  
  extension of algorithm not too difficult

- **nominals (or nominals)** $\rightarrow SHIQO$
  
  e.g., $\text{Human} \sqcap \exists \text{met.Pope}$
  
  extension of algorithm very difficult

Properties of $SHIQO$

- decidable — not yet proven (but there are good reasons)

- **no tree model property**: makes reasoning more difficult!

- **more complex** than $SHIQ$
### Complexity of DLs: Summary

Deciding satisfiability (or subsumption) of

<table>
<thead>
<tr>
<th>concepts in</th>
<th>Definition</th>
<th>without a TBox is</th>
<th>w.r.t. a TBox is</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{ALC}$</td>
<td>$\sqcap, \sqcup, \neg, \exists R.C, \forall R.C,$</td>
<td>PSpace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$S$</td>
<td>$\mathcal{ALC}$ + transitive roles</td>
<td>PSPace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SI$</td>
<td>$SI$ + inverse roles</td>
<td>PSPace-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SH$</td>
<td>$S$ + role hierarchies</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SHIQ$</td>
<td>$SHI$ + number restrictions</td>
<td>ExpTime-c</td>
<td>ExpTime-c</td>
</tr>
<tr>
<td>$SHIQO$</td>
<td>$SHI$ + nominals</td>
<td>NExpTime-c</td>
<td>NExpTime-c</td>
</tr>
<tr>
<td>$SHIQ^+$</td>
<td>$SHIQ$ + “naive number restrictions”</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>$SH^+$</td>
<td>$SH$ + “naive role hierarchies”</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>
The NExpTime tableau algorithm for $\text{SAT}(\text{ALC}, \emptyset)$ can be modified easily to run in PSpace:

For an ALC-concept $C_0$,

1. the c-tree can be built depth-first
2. branches are independent $\implies$ keep only one branch in memory at any time
3. length of branch $\leq |C_0|$
4. for each node $x$, $\mathcal{L}(x) \subseteq \text{sub}(C_0)$ and $\# \text{sub}(C_0)$ is linear in $|C_0|$

$\implies$ non-deterministic PSpace decision procedure for $\text{CSAT}(\text{ALC})$ and Savitch: PSpace = NPSpace
Adding TBoxes to $\mathcal{ALC}$ yields ExpTime-hardness

Why is reasoning w.r.t. TBoxes more complex, i.e., ExpTime-hard?

Intuitively: we can enforce paths of exponential length, i.e.,

there are $C$, $T$ such that, in each model $\mathcal{I}$ of $C$ and $T$, there is a path $x_1, \ldots, x_n$ with $(x_i, x_{i+1}) \in R^\mathcal{I}$ and $n \geq 2^{(|C|+|T|)^2}$

$C$ and $T$ represent binary incrementation using $k$ bits
i-th bit is coded in concept name $X_i$ ($X_k$ is lowest bit, $C \Rightarrow D$ short for $\neg C \sqcup D$)

\[
A = \neg X_1 \sqcap \neg X_2 \sqcap \ldots \sqcap \neg X_k \\
T = \{ A \sqsubseteq \exists R.A \}
\]

\[
A \sqsubseteq (X_k \Rightarrow \forall R.\neg X_k) \sqcap (\neg X_k \Rightarrow \forall R.X_k)
\]

for $i < k$:

\[
\bigoplus_{j < i} X_j \sqsubseteq (X_i \Rightarrow \forall R.\neg X_i) \sqcap (\neg X_i \Rightarrow \forall R.X_i)
\]

\[
\bigoplus_{j < i} \neg X_j \sqsubseteq (X_i \Rightarrow \forall R.X_i) \sqcap (\neg X_i \Rightarrow \forall R.\neg X_i)\}
\]
Adding TBoxes to $\mathcal{ALC}$ yields ExpTime-hardness

Why is reasoning w.r.t. TBoxes more complex, i.e., ExpTime-hard?

**Lemma:** Satisfiability of $\mathcal{ALC}$ w.r.t. TBoxes can be reduced to the Halting Problem of polynomial-space-bounded alternating Turing machines.

We know: the HP-f-PSB-A-TM is ExpTime-hard

**Proof of Lemma:** beyond the scope of this tutorial, but not difficult
**Complexity of SHIQ**

*SHIQ* is ExpTime-hard because *ALC* with TBoxes is and *SHIQ* can internalise TBoxes: polynomially reduce $\text{SAT}(C, T)$ to $\text{SAT}(C_T, \emptyset)$

$$C_T := C \cap \bigcap_{C_i \subseteq D_i \in T} (C_i \Rightarrow D_i) \cap \forall U. \bigcap_{C_i \subseteq D_i \in T} (C_i \Rightarrow D_i)$$

for $U$ new role with $\text{trans}(U)$, and

$$R \subseteq U, R^- \subseteq U$$

for all roles $R$ in $T$ or $C$

**Lemma:** $C$ is satisfiable w.r.t. $T$ iff $C_T$ is satisfiable

Why is *SHIQ* in ExpTime?

Tableau algorithms runs in worst-case non-deterministic double exponential space using double exponential time....
Translation of \texttt{SHIQ} into Büchi Automata on infinite trees

\[ C, T \leadsto A_{C,T} \]

such that

1. \texttt{SAT}(C, T) iff \( L(A_{C,T}) \neq \emptyset \)
2. \(|A_{C,T}|\) is exponential in \(|C| + |T|\)
   (states of \(C,T\) are sets of subconcepts of \(C\) and \(T\))

This yields ExpTime decision procedure for \texttt{SAT}(C, T) since

emptiness of \(L(A)\) can be decided in time polynomial in \(|A|\)

\textbf{Problem} \(A_{C,T}\) needs (?) to be constructed before being tested: best-case ExpTime
**FACT:** for $SHIQ$ and $SHOQ$, $SAT(C,T)$ are ExpTime-complete

$SHOQ$ is $SHIQ$ without inverse roles, with nominals

**Lemma:** their combination is $NExpTime$-hard

even for $ALCQIO$, $SAT(C,T)$ is $NExpTime$-hard

**Proof:** by reduction of a $NExpTime$ version of the domino problem:

can we tile a $2^n \times 2^n$ square using $D$?
NExpTime DLs: *ALCQIO* is NExpTime-hard

**Definition:** A domino system \( \mathcal{D} = (\mathcal{D}, H, V) \)

- set of domino types \( \mathcal{D} = \{D_1, \ldots, D_d\} \), and
- horizontal and vertical matching conditions \( H \subseteq D \times D \) and \( V \subseteq D \times D \)

A tiling of the \( \mathbb{IN} \times \mathbb{IN} \) grid using \( \mathcal{D} \):

\[
\begin{align*}
t : \mathbb{IN} \times \mathbb{IN} & \rightarrow D \text{ such that} \\
\langle t(m, n), t(m + 1, n) \rangle & \in H \text{ and} \\
\langle t(m, n), t(m, n + 1) \rangle & \in V
\end{align*}
\]

**Domino problem**

- standard: has \( \mathcal{D} \) a tiling? undecidable
- exponential: has \( \mathcal{D} \) a tiling for a \( 2^n \times 2^n \) square? NExpTime-c.
Reducing the NExpTime domino problem to $\text{CSAT}(\text{ALCQIO}) \rightsquigarrow$ four tasks:

1. each object carries exactly one domino type $D_i$
   \[ \rightsquigarrow \text{use concept name } D_i \text{ for each domino type and} \]
   \[ T \sqsubseteq \bigcup_{1 \leq i \leq d} (D_i \land \bigcap_{j \neq i} \neg D_j) \]

2. each element $x$ has exactly one $H$-successor
   exactly one $V$-successor
   whose domino types satisfy the horizontal/vertical matching conditions:
   \[ T \sqsubseteq \bigcap_{1 \leq i \leq n} \left( D_i \Rightarrow \left( (\leq_{1V} T) \land \left( \exists V. \bigcup_{(D_i,D_j) \in V} D_j \right) \right) \right) \]
   \[ \left( (\leq_{1H} T) \land \left( \exists H. \bigcup_{(D_i,D_j) \in H} D_j \right) \right) \]
the model must be large enough, i.e., have $2^n \times 2^n$ elements

encode the position $(x, y)$ of each point using binary coding in

the concept names $X_1, \ldots, X_n$, $Y_1, \ldots, Y_n$:

$$
\begin{align*}
\top & \subseteq \exists H. \top \cap \exists V. \top \\
\top & \subseteq (X_k \Rightarrow \forall R. \neg X_k) \cap (\neg X_k \Rightarrow \forall R. X_k) \cap \text{(same for } Y_i) \\
\text{for } i < k : & \quad \bigsqcup_{j < i} X_j \subseteq (X_i \Rightarrow \forall R. \neg X_i) \cap (\neg X_i \Rightarrow \forall R. X_i) \cap \text{(same for } Y_i) \\
\bigsqcup_{j < i} \neg X_j & \subseteq (X_i \Rightarrow \forall R. X_i) \cap (\neg X_i \Rightarrow \forall R. \neg X_i) \cap \text{(same for } Y_i)
\end{align*}
$$

E.g., if $x \in (\neg X_1 \cap X_2 \cap X_3 \cap Y_1 \cap \neg Y_2 \cap Y_3)^\top$, then

$x$ represents $(011, 101)$, and thus the point $(3, 5)$
ensure that the $V \circ H$-successor of each node coincides with its $H \circ V$-successor

\( \implies \) enforce that each object is the $H$-successor of at most one element (and the same for $V$):

\[
T \equiv (\leq 1 V^- . T) \cap (\leq 1 H^- . T)
\]

\( \implies \) enforce that there is \( \leq 1 \) object in the upper right corner:

\[
X_1 \cap \ldots \cap X_n \cap Y_1 \cap \ldots \cap Y_n \subseteq N
\]

for nominal $N$

\[\text{Harvest:} \]

\[
\neg X_1 \cap \ldots \cap \neg X_n \cap \neg Y_1 \cap \ldots \cap \neg Y_n
\]

is satisfiable w.r.t. to $T_D$ defined above iff $D$ has a $2^n \times 2^n$-tiling
In *SHIQ*, each role $R$ in a number restriction ($\forall n R; C$) must be simple, i.e., not $(\exists S)$ for any sub-role $S$ of $R$.

Without this restriction, *SHIQ* (better: *SHQ*) becomes undecidable.

**Proof** by a reduction of the standard, unbounded domino problem.
Remember 4 tasks in the previous domino reduction:

① each object carries exactly one domino type $D_i$
   use concept name $D_i$ for each domino type and

$$
\top \sqsubseteq \bigcup_{1 \leq i \leq d} (D_i \cap \bigcap_{j \neq i} \lnot D_j)
$$

② each element $x$ has exactly one $H$-successor
   exactly one $V$-successor
   whose domino types satisfy the horizontal/vertical matching conditions:

$$
\top \sqsubseteq \bigcap_{1 \leq i \leq n} \left( D_i \Rightarrow ((\leq 1V. \top) \cap (\exists V. \bigcup_{(D_i,D_j) \in V} D_j)) \cap \\
((\leq 1H. \top) \cap (\exists H. \bigcup_{(D_i,D_j) \in H} D_j)) \right)
$$
An Undecidable Extension for \( SHIQ \)

Remember 4 tasks in the previous domino reduction:

3. **model must be large enough**
   \[ \mathcal{T} \subseteq \exists V. \mathcal{T} \cap \exists H. \mathcal{T} \]

4. **vertical-horizontal and horizontal-vertical successor coincide**

- use additional roles \( V_1, V_2 \subseteq V \), \( V_1, V_2 \subseteq V \)
  with additional GCIs, e.g.,
  \[ \mathcal{T} \subseteq (\exists V_1. \mathcal{T} \cap \forall V_1. \forall V_1. \bot) \cup \ldots \]
- **transitive roles** \( D_{i,j} \) with \( H_i, V_j \subseteq D_{i,j} \)
- **number restrictions**
  \[ \mathcal{T} \subseteq \bigcap_{i,j} (\leq 3 \ D_{i,j}, \mathcal{T}) \]
Implementing the *SHIQ* Tableau Algorithm

Naive implementation of *SHIQ* tableau algorithm is doomed to failure:

Construct a tree of exponential depth in a non-deterministic way
\[\rightsquigarrow\] requires backtracking in a deterministic implementation

Optimisations are crucial
  concern every aspect of the help in “many” cases (which?)

In the following: a selection of some vital optimisations
Optimising the \emph{SHIQ} Tableau Algorithm

FaCT provides service "classify all concepts defined \( T \)", i.e.,
for all concept names \( C, D \) defined in \( T \), FaCT decides whether \( C \sqsubseteq_T D \) and \( D \sqsubseteq_T C \)
\[ \sim \text{SAT}(C \sqcap \neg D, T) \text{ and } \text{SAT}(D \sqcap \neg C, T) \]
\[ \sim n^2 \text{ satisifiability tests!} \]

\textbf{Goal: reduce number of satisifiability tests when classifying TBox}

\textbf{Idea:} trickle new concept into hierarchy
computed so far

\begin{center}
\begin{tikzpicture}
\begin{scope}[every node/.style={circle,draw,fill=white,minimum size=1cm}]
  \node (T) at (0,0) {$T$};
  \node (D1) at (1,0) {$D_1$};
  \node (D2) at (3,0) {$D_2$};
  \node (Ci) at (-2,-1) {$C$};
  \node (Di) at (-2,1) {$D_i$};
  \node (Subs) at (-2,0) {$\text{SUBS}(C, D_i, T)$?};
\end{scope}
\end{tikzpicture}
\end{center}
FaCT provides service \textit{“classify all concepts defined }T\textit{”}, i.e.,
for all concept names $C, D$ defined in $T$, FaCT decides whether $C \sqsubseteq_T D$ and $D \sqsubseteq_T C$

$\implies \text{SAT}(C \sqcap \neg D, T)$ and $\text{SAT}(D \sqcap \neg C, T)$

$\implies n^2$ satisfiability tests!

\textbf{Goal: reduce number of satisfiability tests when classifying }T\textbf{-Box}

\textbf{Idea:} trickle new concept into hierarchy computed so far
FaCT provides service “classify all concepts defined $\mathcal{T}$”, i.e., for all concept names $C, D$ defined in $\mathcal{T}$, FaCT decides whether $C \sqsubseteq_\mathcal{T} D$ and $D \sqsubseteq_\mathcal{T} C$

\[ \leadsto \text{SAT}(C \sqcap \neg D, \mathcal{T}) \text{ and } \text{SAT}(D \sqcap \neg C, \mathcal{T}) \]

\[ \leadsto n^2 \text{ satisfiability tests!} \]

**Goal:** reduce number of satisfiability tests when classifying $TBox$

**Idea:** “trickle” new concept into hierarchy

computed so far
Remember: $\rightarrow_{\text{GCI}}$: If $\neg C_i \cup D_i \not\in \mathcal{L}(x)$ for some $1 \leq i \leq n$

Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{(\neg C_i \cup D_i)\}$

Problem: 1 disjunction per GCI $\Rightarrow$ high degree of non-determinism

huge search space

Observation: many GCIs are of the form $A \sqcap \ldots \sqsubseteq C$ for concept name $A$
e.g., $\text{Human} \sqcap \ldots \sqsubseteq C$ versus $\text{Device} \sqcap \ldots \sqsubseteq C$

Idea: restrict applicability of $\rightarrow_{\text{GCI}}$ by translating

$A \sqcap X \sqsubseteq C$ into equivalent $A \sqsubseteq \neg X \cup C$
e.g., $\text{Human} \sqcap \exists\text{owns.Pet} \sqsubseteq C$ becomes $\text{Human} \sqsubseteq \neg \exists\text{owns.Pet} \cup C$

this yields localisation of GCIs to $A$s
Optimising the SHIQ Tableau Algorithm

For SHIQ, the blocking condition is:

\[ y \text{ is blocked by } y' \text{ if} \]

for \( x \) the predecessor of \( y \), \( x' \) the predecessor of \( y' \)

1. \( \mathcal{L}(x) = \mathcal{L}(x') \)
2. \( \mathcal{L}(y) = \mathcal{L}(y') \)
3. \((x, R, y) \iff (x', R, y')\)

\( \rightsquigarrow \text{ blocking occurs late} \)
\( \rightsquigarrow \text{ search space if huge} \)
For \textit{SHIQ}, the blocking condition is:

\[ \text{\textit{y is blocked by \textit{y}}') if} \]

for \( x \) the predecessor of \( y \), \( x' \) the predecessor of \( y' \)

1. \( L(x) = L(x') \)
2. \( L(y) = L(y') \)
3. \((x, R, y) \text{ iff } (x', R, y') \)

\[ \text{for “relevant concepts RC”} \]

\[ \rightsquigarrow \text{ blocking occurs late} \]
\[ \rightsquigarrow \text{ search space if huge} \]

1. \( L(x) \cap RC = L(x') \cap RC \)
2. \( L(y) \cap RC = L(y') \cap RC \)
3. \((x, R, y) \text{ iff } (x', R, y') \)

\[ \rightsquigarrow \text{ blocking occurs earlier} \]
\[ \rightsquigarrow \text{ search space if smaller} \]
Remember If a clash \((A, \neg A \in \mathcal{L}(x))\) is encountered, algorithm backtracks
i.e., returns to last non-deterministic choice and
tries other possibility

Example \(\exists R. (A \cap B) \cap (C_1 \cup D_1) \cap \ldots \cap (C_1 \cup D_1) \cap \forall R. \neg A \in \mathcal{L}(x)\)
Remember If a clash \((A, \neg A \in \mathcal{L}(x))\) is encountered, algorithm backtracks i.e., returns to last non-deterministic choice and tries other possibility

Example \(\exists R. (A \cap B) \cap (C_1 \cup D_1) \cap \ldots \cap (C_1 \cup D_1) \cap \forall R. \neg A \in \mathcal{L}(x)\)
Remember If a clash \((A, \neg A \in \mathcal{L}(x))\) is encountered, algorithm backtracks

i.e., returns to last non-deterministic choice and
tries other possibility

Example \(\exists R.(A \cap B) \cap (C_1 \cup D_1) \cap \ldots \cap (C_1 \cup D_1) \cap \forall R.\neg A \in \mathcal{L}(x)\)
Remember If a clash \((A, \neg A \in \mathcal{L}(x))\) is encountered, algorithm backtracks

i.e., returns to last non-deterministic choice and
tries other possibility

Example \(\exists R. (A \cap B) \cap (C_1 \cup D_1) \cap \ldots \cap (C_1 \cup D_1) \cap \forall R. \neg A \in \mathcal{L}(x)\)
Finally: *SHIQ* extends *propositional logic*

\[\sim \text{heuristics developed for SAT are relevant}\]

Summing up: optimisations at each aspect of tableau algorithm can dramatically enhance performance

\[\sim \text{do they interact?}\]

\[\sim \text{how?}\]

\[\sim \text{which combination works best for which “cases”?}\]

\[\sim \text{is the optimised algorithm still correct?}\]
5. Future Challenges, Outlook, and Leftovers

Ian Horrock and Ulrike Sattler
University of Manchester
Manchester, UK
{horrocks|sattler}@cs.man.ac.uk
Plan for today

1. ABoxes and instances
2. “non-standard” reasoning services
3. Nominals
4. Propagation
5. Concrete Domains
6. Keys
7. uuups - I get carried away
Remember: when using ontologies, we would like to automatically classify individuals described in an ABox

an ABox $\mathcal{A}$ is a finite set of assertions of the form

$$C(a) \text{ or } R(a, b)$$

How to decide whether $\text{Inst}(a, \mathcal{A}, T)$? I.e., whether $a \in C^I$ in all models $I$ of $T$?

$\rightsquigarrow$ extend tableau algorithm to start with ABox $C(a) \in \mathcal{A} \Rightarrow C \in \mathcal{L}(a)$

$R(a, b) \in \mathcal{A} \Rightarrow (a,R,y)$

work on forest (rather than on a single tree)

i.e., trees whose root nodes intertwine

theoretically not too complicated

many problems in implementation
For Ontology Engineering, useful reasoning services can be based on SAT and SUBS.

Are all useful reasoning services based on SAT and SUBS?

Remember: to support modifying ontologies, we wanted

- automatic generation of concept definitions from examples
  - given ABox $\mathcal{A}$ and individuals $a_i$ create
    a (most specific) concept $C$ such that each $a_i \in C^\mathcal{I}$ in each model $\mathcal{I}$ of $\mathcal{T}$
    $$\text{msc}(a_1, \ldots, a_n), \mathcal{A}, \mathcal{T}$$

- automatic generation of concept definitions for too many siblings
  - given concepts $C_1, \ldots, C_n$, create
    a (most specific) concept $C$ such that $\text{SUBS}(C_i, C, \mathcal{T})$
    $$\text{lcs}(C_1, \ldots, C_n), \mathcal{A}, \mathcal{T}$$
Unlike SAT, SUBS, etc., msc is a computation problem (not decision problem)

Idea: \[ \text{msc}(a_1, \ldots, a_n, \mathcal{A}, \mathcal{T}) = \text{lcs}(\text{msc}(a_1, \mathcal{A}, \mathcal{T}), \ldots, \text{msc}(a_n, \mathcal{A}, \mathcal{T})) \]

Known Results:

- lcs in DLs with \( \sqsubseteq \) is useless
- \( \text{msc}(a_1, \mathcal{A}, \mathcal{T}) \) does not need to exist
Acknowledgements

Thanks to various people from whom I “borrowed” material:

- Jeen Broekstra
- Carole Goble
- Frank van Harmelen
- Austin Tate
- Raphael Volz

And thanks to all the people from whom they borrowed it 😊
Intelligent Tools Demo
Resources

- Course material (including slides, tools and ontologies):

- Description Logic Handbook
  - http://books.cambridge.org/0521781760.htm
Additional Material
Tableau rules for $\mathcal{ALC}$

$\rightarrow_\cap$: If $C \cap D \in \mathcal{L}(x)$ but $\{C, D\} \cap \mathcal{L}(x) = \emptyset$
Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C, D\}$

$\rightarrow_\cup$: If $C \cup D \in \mathcal{L}(x)$ but $\{C, D\} \not\subseteq \mathcal{L}(x)$
Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$ for $E \in \{C, D\}$

$\rightarrow_\exists$: If $\exists R.C \in \mathcal{L}(x)$ but $x$ has no $R$-successor $y$ with $C \in \mathcal{L}(y)$
Then create new $R$-successor $y$ of $x$ with
$\mathcal{L}(y) = \{C\}$

$\rightarrow_\forall$: If $\forall R.C \in \mathcal{L}(x)$ and $x$ has an $R$-successor $y$ with $C \not\in \mathcal{L}(y)$
Then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$
Tableau rules for $ALC$ with GCIs

$$\{C_i \subseteq D_i \mid 1 \leq i \leq n\}$$

applicable only to nodes $x$ that are not blocked:

**y is blocked by an ancestor $x$ if $\mathcal{L}(y) \subseteq \mathcal{L}(x)$**

$\rightarrow_\cap$: If $C \cap D \in \mathcal{L}(x)$ but $\{C, D\} \cap \mathcal{L}(x) = \emptyset$
Then $\mathcal{L}(x) \to \mathcal{L}(x) \cup \{C, D\}$

$\rightarrow_\cup$: If $C \cup D \in \mathcal{L}(x)$ but $\{C, D\} \not\subseteq \mathcal{L}(x)$
Then $\mathcal{L}(x) \to \mathcal{L}(x) \cup \{E\}$ for $E \in \{C, D\}$

$\rightarrow_\exists$: If $\exists R.C \in \mathcal{L}(x)$ but $x$ has no $R$-successor $y$
with $C \in \mathcal{L}(y)$
Then create new $R$-successor $y$ of $x$ with
$\mathcal{L}(y) = \{C\}$

$\rightarrow_\forall$: If $\forall R.C \in \mathcal{L}(x)$ and $x$ has an $R$-successor $y$
with $C \notin \mathcal{L}(y)$
Then $\mathcal{L}(y) \to \mathcal{L}(y) \cup \{C\}$

$\rightarrow_{GCI}$: If $(\neg C_i \sqcup D_i) \not\subseteq \mathcal{L}(x)$
for some $1 \leq i \leq n$
Then $\mathcal{L}(x) \to \mathcal{L}(x) \cup \{-C_i \sqcup D_i\}$
Tableau rules for $\mathcal{ALCI}$ with GCIs

$$\{C_i \sqsubseteq D_i \mid 1 \leq i \leq n\}$$

applicable only to nodes $x$ that are not blocked:

$y$ is blocked by an ancestor $x$ if $\mathcal{L}(x) = \mathcal{L}(y)$

$\rightarrow_\forall$: If $C \cap D \in \mathcal{L}(x)$ but $\{C, D\} \cap \mathcal{L}(x) = \emptyset$

Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C, D\}$

$\rightarrow_\forall$: If $C \cup D \in \mathcal{L}(x)$ but $\{C, D\} \not\subseteq \mathcal{L}(x)$

Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$ for $E \in \{C, D\}$

$\rightarrow_\exists$: If $\exists R.C \in \mathcal{L}(x)$ but $x$ has no $R$-neighbour $y$ with $C \in \mathcal{L}(y)$

Then create new $R$-successor $y$ of $x$ with $\mathcal{L}(y) = \{C\}$

$\rightarrow_\forall$: If $\forall R.C \in \mathcal{L}(x)$ and $x$ has an $R$-neighbour $y$ with $C \notin \mathcal{L}(y)$

Then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$

$\rightarrow_{\text{GCI}}$: If $(\neg C_i \sqcup D_i) \notin \mathcal{L}(x)$ for some $1 \leq i \leq n$

Then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_T\}$
Additional tableau rules for ALCQI with GCIs applicable only to nodes $x$ that are not blocked:

$y$ is blocked by an ancestor $y'$ if there are $x$, $x'$ with

- $y$ is succ. of $x$, $y'$ is succ. of $x'$,
- $\mathcal{L}(x) = \mathcal{L}(y)$, $\mathcal{L}(x') = \mathcal{L}(y')$, and
- $\mathcal{L}(\langle x, y \rangle) = \mathcal{L}(\langle x', y' \rangle)$.

$\rightarrow_{\geq}$: If $(\geq nR.C) \in \mathcal{L}(x), x$ is not blocked, and $x$ has less than $n$ $R$-neighbours $y_i$ with $C \in \mathcal{L}(y_i)$

Then create $n$ new $R$-successor $y_1, \ldots, y_n$ of $x$ with

$\mathcal{L}(y_i) := \{C\}$ and $y_i \neq y_j$ for all $i \neq j$

$\rightarrow_{\leq}$: If $(\leq nR.C) \in \mathcal{L}(x), x$ is not indirectly blocked, $x$ has $n+1$ $R$-neighbours $y_0, \ldots, y_n$ with $C \in \mathcal{L}(y_i)$, and there are $i, j$ with not $y_i \neq y_j$ and $y_j$ is not an ancestor of $y_i$

Then $\mathcal{L}(y_i) \rightarrow \mathcal{L}(y_i) \cup \mathcal{L}(y_j)$,

make $y_j$'s successors to successors of $y_i$,

add $y_i \neq z$ for each $z$ with $y_j \neq z$,

remove $y_j$ from the tree

$\rightarrow_{\text{choice}}$: If $(\leq nR.C) \in \mathcal{L}(x), x$ is not indirectly blocked, $x$ has an $R$-neighbour $y$ with

$\{C, \neg C\} \cap \mathcal{L}(y) = \emptyset$

Then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{D\}$ for some $D \in \{C, \neg C\}$
Translation of \( ALCQI\O\) -concepts into C2

(The mapping \( t_y \) is obtained by switching the roles of \( x \) and \( y \) in \( t_x \))

\[
\begin{align*}
t_x(A) &= A(x), \\
t_x(\neg C) &= \neg t_x(C) \\
t_x(C \cap D) &= t_x(C) \land t_x(D), \\
t_x(C \cup D) &= t_x(C) \lor t_x(D), \\
t_x(\exists R.C) &= \exists y. R(x, y) \land t_y(C) \\
t_x(\forall R.C) &= \forall y. \neg R(x, y) \lor t_y(C) \\
t_x(\geq n R.C) &= \exists y. R(x, y) \land t_y(C), \\
t_x(\geq n R^- . C) &= \exists y. R(y, x) \land t_y(C), \\
t_x(\leq n R.C) &= \exists y. R(x, y) \land t_y(C), \\
t_x(\leq n R^- . C) &= \exists y. R(y, x) \land t_y(C)
\end{align*}
\]

\[
t(T) = \bigwedge_{C \subseteq D \in T} \forall x. t_x(C) \Rightarrow t_x(D)
\]

\[
t(R \sqsubseteq S) = \forall x, y. R(x, y) \Rightarrow S(x, y)
\]

\[
t(\text{trans}(R)) = \forall x, y, z. (R(x, y) \land R(y, z)) \Rightarrow R(x, z)
\]

\[
t_x(o) = (x = a_o), \text{ for nominal } o \text{ and constant } a_o
\]

Lemma:

1. \( \text{sat}(C, T) \iff t_x(C) \land t(T) \text{ is satisfiable} \)
2. \( \text{sat}(C, D, T) \iff t(T) \Rightarrow (\forall x. t_x(C) \Rightarrow t_x(D)) \)