Ian Horrocks  
Information Management Group  
University of Manchester, UK

Ulrike Sattler  
Institut für Theoretische Informatik  
TU Dresden, Germany
Overview of the Tutorial

• **History and Basics:** Syntax, Semantics, ABoxes, Tboxes, Inference Problems and their interrelationship, and Relationship with other (logical) formalisms

• **Applications of DLs:** ER-diagrams with i.com demo, ontologies, etc. including system demonstration

• **Reasoning Procedures:** simple tableaux and why they work

• **Reasoning Procedures II:** more complex tableaux, non-standard inference problems

• **Complexity issues**

• **Implementing/Optimising DL systems**
Description Logics

- family of logic-based knowledge representation formalisms well-suited for the representation of and reasoning about
  - terminological knowledge
  - configurations
  - ontologies
  - database schemata
    - schema design, evolution, and query optimisation
    - source integration in heterogeneous databases/data warehouses
    - conceptual modelling of multidimensional aggregation
  - ...

- descendents of semantics networks, frame-based systems, and KL-ONE

- aka terminological KR systems, concept languages, etc.
Concrete Situation

Terminology

Knowledge Base

Description Logic

\[ \text{Father} = \text{Man} \sqcap \exists \text{has-child}. T \]
\[ \text{Human} = \text{Mammal} \sqcap \text{Biped} \]

\[ \text{John:Human} \sqcap \text{Father} \]
\[ \text{John has child Bill} \]
A Description Logic - mainly characterised by a set of constructors that allow to build complex concepts and roles from atomic ones,

- concepts correspond to classes / are interpreted as sets of objects,
- roles correspond to relations / are interpreted as binary relations on objects,

Example: Happy Father in the DL \(\mathcal{ALC}\)

\[
\text{Man} \sqcap (\exists \text{has-child}. \text{Blue}) \sqcap (\exists \text{has-child}. \text{Green}) \sqcap (\forall \text{has-child}. \text{Happy} \sqcup \text{Rich})
\]
Semantics given by means of an interpretation $\mathcal{I} = (\Delta^I, \cdot^I)$:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>$A$</td>
<td>Human</td>
<td>$A^I \subseteq \Delta^I$</td>
</tr>
<tr>
<td>atomic role</td>
<td>$R$</td>
<td>likes</td>
<td>$R^I \subseteq \Delta^I \times \Delta^I$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
<td>Human $\sqcap$ Male</td>
<td>$C^I \cap D^I$</td>
</tr>
<tr>
<td>disjunction</td>
<td>$C \sqcup D$</td>
<td>Nice $\sqcup$ Rich</td>
<td>$C^I \cup D^I$</td>
</tr>
<tr>
<td>negation</td>
<td>$\neg C$</td>
<td>$\neg$ Meat</td>
<td>$\Delta^I \setminus C^I$</td>
</tr>
<tr>
<td>exists restrict.</td>
<td>$\exists R.C$</td>
<td>$\exists$ has-child.Human</td>
<td>${ x \mid \exists y. \langle x, y \rangle \in R^I \land y \in C^I }$</td>
</tr>
<tr>
<td>value restrict.</td>
<td>$\forall R.C$</td>
<td>$\forall$ has-child.Blond</td>
<td>${ x \mid \forall y. \langle x, y \rangle \in R^I \Rightarrow y \in C^I }$</td>
</tr>
</tbody>
</table>
### Introduction to DL: Other DL Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>number restriction</td>
<td>$(\geq n \ R)$</td>
<td>$(\geq 7 \ has\text{-}child)$</td>
<td>${x \mid</td>
</tr>
<tr>
<td></td>
<td>$(\leq n \ R)$</td>
<td>$(\leq 1 \ has\text{-}mother)$</td>
<td>${x \mid</td>
</tr>
<tr>
<td>inverse role</td>
<td>$R^-$</td>
<td>has-child$^-$</td>
<td>${\langle x, y \rangle \mid \langle y, x \rangle \in R^T}$</td>
</tr>
<tr>
<td>trans. role</td>
<td>$R^*$</td>
<td>has-child$^*$</td>
<td>$(R^T)^*$</td>
</tr>
<tr>
<td>concrete domain</td>
<td>$u_1, \ldots, u_n.P$</td>
<td>h-father·age, age. $&gt;$</td>
<td>${x \mid \langle u_1^T, \ldots, u_n^T \rangle \in P}$</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Many different DLs/DL constructors have been investigated.
Introduction to DL: Knowledge Bases: TBoxes

**For terminological knowledge:** TBox contains

**Concept definitions**

\[ A \triangleq C \]  (\( A \) a concept name, \( C \) a complex concept)

\[
\begin{align*}
\text{Father} & \triangleq \text{Man} \sqcap \exists \text{has-child}.\text{Human} \\
\text{Human} & \triangleq \text{Mammal} \sqcap \forall \text{has-child}^-.\text{Human}
\end{align*}
\]

\( \sim \) introduce macros/names for concepts, can be (a)cyclic

**Axioms**

\[ C_1 \sqsubseteq C_2 \]  (\( C_i \) complex concepts)

\[
\begin{align*}
\exists \text{favourite}.\text{Brewery} & \sqsubseteq \exists \text{drinks}.\text{Beer} \\
\sim & \text{restrict your models}
\end{align*}
\]

**An interpretation** \( \mathcal{I} \) satisfies

- a concept definition \( A \triangleq C \) iff \( A^\mathcal{I} = C^\mathcal{I} \)

- an axiom \( C_1 \sqsubseteq C_2 \) iff \( C_1^\mathcal{I} \subseteq C_2^\mathcal{I} \)

- a TBox \( \mathcal{T} \) iff \( \mathcal{I} \) satisfies all definitions and axioms in \( \mathcal{T} \)

\( \sim \) \( \mathcal{I} \) is a model of \( \mathcal{T} \)
For assertional knowledge: ABox contains

Concept assertions  
\[ a : C \]  \( a \) an individual name, \( C \) a complex concept)  
\[ \text{John} : \text{Man} \sqcap \forall \text{has-child.}(\text{Male} \sqcap \text{Happy}) \]  

Role assertions  
\[ \langle a_1, a_2 \rangle : R \]  \( a_i \) individual names, \( R \) a role)  
\[ \langle \text{John}, \text{Bill} \rangle : \text{has-child} \]

An interpretation \( \mathcal{I} \) satisfies

a concept assertion  
\[ a : C \text{ iff } a^\mathcal{I} \in C^\mathcal{I} \]

a role assertion  
\[ \langle a_1, a_2 \rangle : R \text{ iff } \langle a_1^\mathcal{I}, a_2^\mathcal{I} \rangle \in R^\mathcal{I} \]

an ABox  
\[ \mathcal{A} \text{ iff } \mathcal{I} \text{ satisfies all assertions in } \mathcal{A} \]
\[ \mathcal{I} \text{ is a model of } \mathcal{A} \]
**Subsumption:** \( C \sqsubseteq D \)  
Is \( C^I \sqsubseteq D^I \) in all interpretations \( I \)?

w.r.t. TBox \( T \): \( C \sqsubseteq_T D \)  
Is \( C^I \sqsubseteq D^I \) in all models \( I \) of \( T \)?

\( \sim \) structure your knowledge, compute taxonomy

**Consistency:** Is \( C \) consistent w.r.t. \( T \)? Is there a model \( I \) of \( T \) with \( C^I \neq \emptyset \)?

- of ABox \( A \): Is \( A \) consistent? Is there a model of \( A \)?
- of KB \( (T,A) \): Is \( (T,A) \) consistent? Is there a model of both \( T \) and \( A \)?

**Inference Problems are closely related:**

\[ C \sqsubseteq_T D \iff C \sqcap \neg D \text{ is inconsistent w.r.t. } T, \]

\( \text{(no model of } I \text{ has an instance of } C \sqcap \neg D) \)

\[ C \text{ is consistent w.r.t. } T \iff \text{not } C \sqsubseteq_T A \sqcap \neg A \]

\( \sim \) Decision Procedures for consistency (w.r.t. TBoxes) suffice
For most DLs, the basic inference problems are decidable, with complexities between $P$ and $\text{ExpTime}$.

Why is decidability important? Why does semi-decidability not suffice?

If subsumption (and hence consistency) is undecidable, and

- subsumption is semi-decidable, then consistency is not semi-decidable
- consistency is semi-decidable, then subsumption is not semi-decidable

- Quest for a “highly expressive” DL with “practicable” inference problems

  where expressiveness depends on the application
  practicability changed over the time
Complexity of Inferences provided by DL systems over the time

Investigation of Complexity of Inference Problems/Algorithms starts

- Undecidable
  - KL-ONE
  - NIKL

- ExpTime
  - Loom

- PSpace
  - Crack, Kris

- NP

- PTime
  - Classic (AT&T)

- late '80s
- early '90s
- mid '90s
- late '90s

TU Dresden
Germany

12
In the last 5 years, DL-based systems were built that

✔ can handle DLs far more expressive than $\mathcal{ALC}$ (close relatives of converse-DPDL)
  • Number restrictions: “people having at most 2 cats and exactly 1 dog”
  • Complex roles: inverse (“has-child” → “child-of”),
    transitive closure (“offspring” → “has-child”),
    role inclusion (“has-daughter” → “has-child”), etc.

✔ implement provably sound and complete inference algorithms
  (for ExpTime-complete problems)

✔ can handle large knowledge bases
  (e.g., Galen medical terminology ontology: 2,740 concepts, 413 roles, 1,214 axioms)

✔ are highly optimised versions of tableau-based algorithms

✔ perform (surprisingly well) on benchmarks for modal logic reasoners
  (Tableaux’98, Tableaux’99)
Most DLs are decidable fragments of FOL: Introduce

a unary predicate $A$ for a concept name $A$
a binary relation $R$ for a role name $R$

Translate complex concepts $C, D$ as follows:

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

A TBox $T = \{C_i \models D_i\}$ is translated as

\[
\Phi_T = \forall x. \bigwedge_{1 \leq i \leq n} t_x(C_i) \Leftrightarrow t_x(D_i)
\]
$C$ is consistent iff its translation $t_x(C)$ is satisfiable,

$C$ is consistent w.r.t. $T$ iff its translation $t_x(C) \land \Phi_T$ is satisfiable,

$C \sqsubseteq D$ iff $t_x(C) \Rightarrow t_x(D)$ is valid

$C \sqsubseteq_T D$ iff $\Phi_t \Rightarrow \forall x. (t_x(C) \Rightarrow t_x(D))$ is valid.

$\rightsquigarrow \mathcal{ALC}$ is a fragment of FOL with 2 variables (L2), known to be decidable

$\rightsquigarrow \mathcal{ALC}$ with inverse roles and Boolean operators on roles is a fragment of L2

$\rightsquigarrow$ further adding number restrictions yields a fragment of C2

(L2 with “counting quantifiers”), known to be decidable

✧ in contrast to most DLs, adding transitive roles (binary relations/transitive closure operator) to L2 leads to undecidability

✧ many DLs (like many modal logics) are fragments of the Guarded Fragment

✧ most DLs are less complex than L2: L2 is NExpTime-complete, most DLs are in ExpTime
DLs and Modal Logics are closely related:

\[ \mathcal{ALC} \iff \text{multi-modal } K: \]

\[ C \cap D \iff C \land D, \quad C \cup D \iff C \lor D \]
\[ \neg C \iff \neg C, \quad \exists R.C \iff \langle R\rangle C, \quad \forall R.C \iff [R]C \]

- Transitive roles \( \mapsto \) transitive frames (e.g., in K4)
- Regular expressions on roles \( \mapsto \) regular expressions on programs (e.g., in PDL)
- Inverse roles \( \mapsto \) converse programs (e.g., in C-PDL)
- Number restrictions \( \mapsto \) deterministic programs (e.g., in D-PDL)

\( \not\Rightarrow \) no TBoxes available in modal logics
\( \not\Rightarrow \) “internalise” axioms using a universal role \( u: C \models D \iff [u](C \iff D) \)

\( \not\Rightarrow \) no ABox available in modal logics \( \not\Rightarrow \) use nominals
Applications of Description Logics
Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
- Resulting language (DAML+OIL) is **based on a DL** ($SHIQ$)
- DL **reasoning** can be used to, e.g.,
  - Support ontology design and maintenance
  - Classify resources w.r.t. ontologies
Application Areas II

Configuration
- **Classic** system used to configure telecoms equipment
- Characteristics of components described in DL KB
- Reasoner checks validity (and price) of configurations

Software information systems
- LaSSIE system used DL KB for flexible software documentation and query answering

Database applications
Database Schema and Query Reasoning

- $DLR$ (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to $SHIQ$ allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support
  - Schema design, evolution and query optimisation
  - Source integration in heterogeneous databases/data warehouses
  - Conceptual modelling of multidimensional aggregation
- E.g., I.COM Intelligent Conceptual Modelling tool (Enrico Franconi)
  - Uses FaCT system to provide reasoning support for EER
I.COM Demo
Terminological KR and Ontologies

General requirement for medical terminologies

- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions

Galen project aims to replace static hierarchy with DL

- **Describe** concepts (e.g., spiral fracture of left femur)
- Use DL classifier to **build taxonomy**

Needed expressive DL and efficient reasoning

- Descriptions use transitive/inverse roles, GCIs etc.
- Very large KBs (tens of thousands of concepts)
  - Even prototype KB is very large (∼3,000 concepts)
  - Existing (incomplete) classifier took ∼24 hours to classify KB
  - FaCT system (sound and complete) takes ∼60 seconds
Reasoning Support for Ontology Design

DL reasoner can be used to support design and maintenance

Example is OilEd ontology editor (for DAML+OIL)

- Frame based interface (like Protegé, OntoEdit, etc.)
- Extended to clarify semantics and capture whole DAML+OIL language
  - Slots explicitly existential or value restrictions
  - Boolean connectives and nesting
  - Properties for slot relations (transitive, functional etc.)
  - General axioms

Reasoning support for OilEd provided by FaCT system

- Frame representation translated into $SHIQ$
- Communicates with FaCT via CORBA interface
- Indicates inconsistencies and implicit subsumptions
- Can make implicit subsumptions explicit in KB
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture transitive partonomy, causality, etc.
  Smoking ⊑ ∃causes.Cancer plus Cancer ⊑ ∃causes.Death
  ⇒ Cancer ⊑ FatalThing

☞ GCIs represent additional non-definitional knowledge
  Stomach-Ulcer ⊑ Ulcer ⊑ ∃hasLocation.Stomach plus
  Stomach-Ulcer ⊑ ∃hasLocation.Lining-Of-Stomach
  ⇒ Ulcer ⊑ ∃hasLocation.Stomach ⊑ OrganLiningLesion

☞ Inverse roles capture e.g. causes/causedBy relationship
  Death ⊑ ∃causedBy.Smoking ⊑ PrematureDeath
  ⇒ Smoking ⊑ CauseOfPrematureDeath

☞ Cardinality restrictions add consistency constraints
  BloodPressure ⊑ ∃hasValue.(High ⊔ Low) ⊑ ≤1hasValue plus
  High ⊑ ¬Low ⇒ HighLowBloodPressure ⊑ ⊥
As a warm-up, we describe a **tableau-based algorithm** that

- decides consistency of $\mathcal{ALCN}$ concepts,
- tries to build a (tree) model $\mathcal{I}$ for input concept $C_0$,
- breaks down $C_0$ syntactically, inferring constraints on elements in $\mathcal{I}$,
- uses **tableau rules** corresponding to operators in $\mathcal{ALCN}$ (e.g., $\rightarrow \Box$, $\rightarrow \exists$)
- works non-deterministically, in PSpace
- stops when **clash** occurs
- terminates
- returns “$C_0$ is consistent” iff $C_0$ is consistent
Reasoning Procedures: Tableau Algorithm

- works on a tree (semantics through viewing tree as an ABox):
  - **nodes** represent elements of $\Delta^I$, labelled with sub-concepts of $C_0$
  - **edges** represent role-successorships between elements of $\Delta^I$

- works on concepts in **negation normal form**: push negation inside using de Morgan’ laws and

\[
\neg(\exists R.C) \leadsto \forall R.\neg C \\
\neg(\leq n R) \leadsto (\geq (n + 1) R) \\
\neg(\geq n R) \leadsto (\leq (n - 1) R) \quad (n \geq 1) \\
\neg(\geq 0 R) \leadsto A \cap \neg A
\]

- is initialised with a tree consisting of a single (root) node $x_0$ with $\mathcal{L}(x_0) = \{C_0\}$:

- a tree $T$ contains a **clash** if, for a node $x$ in $T$,

\[
\{A, \neg A\} \subseteq \mathcal{L}(x) \quad \text{or} \\
\{(\geq m R), (\leq n R)\} \subseteq \mathcal{L}(x) \quad \text{for} \quad n < m
\]

- returns “$C_0$ is consistent” if rules can be applied s.t. they yield a clash-free, complete (no more rules apply) tree
<table>
<thead>
<tr>
<th>Reasoning Procedures: ( \mathcal{ALC} ) Tableau Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \bullet { C_1 \sqcap C_2, \ldots } )</td>
</tr>
<tr>
<td>( x \bullet { C_1 \sqcup C_2, \ldots } )</td>
</tr>
<tr>
<td>( x \bullet { \exists R.C, \ldots } )</td>
</tr>
<tr>
<td>( x \bullet { \forall R.C, \ldots } )</td>
</tr>
</tbody>
</table>

- for \( C \in \{ C_1, C_2 \} \)
Reasoning Procedures: \( \mathcal{N} \) Tableau Rules

\[
x \bullet \{ (\geq n \ R) \}, \ldots \}; \quad \rightarrow_{\geq} \quad x \bullet \{ (\geq n \ R) \}, \ldots \}
\]

\( x \) has no \( R \)-succ.

\[
x \bullet \{ (\leq n \ R) \}, \ldots \}; \quad \rightarrow_{\leq} \quad x \bullet \{ (\leq n \ R) \}, \ldots \}
\]

R
\[
\ldots > n
\]

\[
x \bullet \{ (\leq n \ R) \}, \ldots \}
\]

merges two \( R \)-succs.

\[
x \bullet \{ (\geq n \ R) \}, \ldots \}
\]

\[
R
\]

\[
\{\}
\]
Lemma Let $C_0$ be an $\mathcal{ALCN}$ concept and $T$ obtained by applying the tableau rules to $C_0$. Then

1. the rule application terminates,
2. if $T$ is clash-free and complete, then $T$ defines (canonical) (tree) model for $C_0$, and
3. if $C_0$ has a model $\mathcal{I}$, then the rules can be applied such that they yield a clash-free and complete $T$.

Corollary

(1) The tableau algorithm is a (PSpace) decision procedure for consistency (and subsumption) of $\mathcal{ALCN}$ concepts

(2) $\mathcal{ALCN}$ has the tree model property
Proof of the Lemma

1. (Termination) The algorithm “monotonically” constructs a tree whose
   depth is linear in $|C_0|$: quantifier depth decreases from node to succs.
   breadth is linear in $|C_0|$ (even if number in NRs are coded binarily)

2. (Canonical model) Complete, clash-free tree $T$ defines a (tree) pre-model $\mathcal{I}$:

   - nodes $x$ correspond to elements $x \in \Delta^\mathcal{I}$
   - edges $x \xrightarrow{R} y$ define role-relationship
   - $x \in A^\mathcal{I}$ iff $A \in \mathcal{L}(x)$ for concept names $A$

   $\leadsto$ Easy to that $C \in \mathcal{L}(x) \Rightarrow x \in C^\mathcal{I}$ — if $C \neq (\geq n \ R)$

   If $(\geq n \ R) \in \mathcal{L}(x)$, then $x$ might have less than $n$ $R$-successors, but
   the $\xrightarrow{\geq}$-rule ensures that there is $\geq 1$ $R$-successor...
copy some $R$-successors (including sub-trees) to obtain $n R$-successors:

$$\sim\text{ canonical tree model for input concept}$$

3. (Completeness) Use model $\mathcal{I}$ of $C_0$ to steer application of non-deterministic rules $(\rightarrow_\sqcup, \rightarrow_\preceq)$ via mapping

$$\pi : \text{Nodes of Tree} \rightarrow \Delta^\mathcal{I} \quad \text{with} \quad C \in \mathcal{L}(x) \Rightarrow \pi(x) \in C^\mathcal{I}.$$ 

This easily implies clash-freeness of the tree generated.
To make the tableau algorithm run in PSpace:

① observe that branches are independent from each other
② observe that each node (label) requires linear space only
③ recall that paths are of length $\leq |C_0|$
④ construct/search the tree depth first
⑤ re-use space from already constructed branches

$\leadsto$ space polynomial in $|C_0|$ suffices for each branch/for the algorithm

$\leadsto$ tableau algorithm runs in NPspace (Savitch: NPspace = PSpace)
This tableau algorithm can be modified to a PSpace decision procedure for

✔ $\mathcal{ALC}$ with qualifying number restrictions
  $(\geq n \ R \ C)$ and $(\leq n \ R \ C)$
✔ $\mathcal{ALC}$ with inverse roles $\text{has-child}^-$
✔ $\mathcal{ALC}$ with role conjunction
  $\exists (R \cap S).C$ and $\forall (R \cap S).C$
✔ TBoxes with acyclic concept definitions:
  unfolding (macro expansion) is easy, but suboptimal:
    may yield exponential blow-up
  lazy unfolding (unfolding on demand) is optimal, consistency in PSpace decidable
Language extensions that require more elaborate techniques include

- **TBoxes with general axioms** $C_i \subseteq D_i$:
  - each node must be labelled with $\neg C_i \cup D_i$
  - quantifier depth no longer decreases
  - $\not\rightarrow$ termination not guaranteed

- **Transitive closure of roles**:
  - node labels $(\forall R^* . C)$ yields $C$ in all $R^n$-successor labels
  - quantifier depth no longer decreases
  - $\not\rightarrow$ termination not guaranteed

Use **blocking** (cycle detection) to ensure termination
(but the right blocking to retain soundness and completeness)
Non-Termination

As already mentioned, for $\mathcal{ALC}$ with **general axioms** basic algorithm is **non-terminating**

E.g. if $\text{human} \sqsubseteq \exists\text{has-mother.}: \text{human} \in \mathcal{T}$, then

$\neg \text{human} \sqcup \exists\text{has-mother.}: \text{human}$ added to every node

\[
\begin{align*}
\mathcal{L}(w) &= \{\text{human}, (\neg \text{human} \sqcup \exists\text{has-mother.}: \text{human}), \exists\text{has-mother.}: \text{human}\} \\
\text{has-mother} \\
\mathcal{L}(x) &= \{\text{human}, (\neg \text{human} \sqcup \exists\text{has-mother.}: \text{human}), \exists\text{has-mother.}: \text{human}\} \\
\text{has-mother} \\
\mathcal{L}(y) &= \{\text{human}, (\neg \text{human} \sqcup \exists\text{has-mother.}: \text{human}), \exists\text{has-mother.}: \text{human}\} \\
\ldots
\end{align*}
\]
Blocking

- When creating new node, check ancestors for equal (superset) label
- If such a node is found, new node is **blocked**

\[
\mathcal{L}(w) = \{ \text{human}, (\neg \text{human} \cup \exists \text{has-mother}.\text{human}), \exists \text{has-mother}.\text{human} \}
\]

\[
\mathcal{L}(x) = \{ \text{human}, (\neg \text{human} \cup \exists \text{has-mother}.\text{human}) \}
\]
Simple subset blocking may not work with more complex logics

E.g., reasoning with inverse roles
  - Expanding node label can affect predecessor
  - Label of blocking node can affect predecessor
  - E.g., testing $C \sqcap \exists S.C$ w.r.t. Tbox

\[
\mathcal{T} = \{ \top \subseteq \forall R^-(\forall S^- \neg C), \top \subseteq \exists R.C \}
\]
Dynamic Blocking

Solution (for inverse roles) is **dynamic blocking**

- Blocks can be established broken and re-established
- Continue to expand $\forall R.C$ terms in blocked nodes
- Check that cycles satisfy $\forall R.C$ concepts

\[
\mathcal{L}(y) = \{ C, \forall \neg R.(\forall \neg S. \neg C), \exists R.C \}\]

\[
\mathcal{L}(w) = \{ C, \exists S.C, \forall \neg R.(\forall \neg S. \neg C), \exists R.C, \forall \neg S. \neg C, \neg C \}\]

\[
\mathcal{L}(x) = \{ C, \forall \neg R.(\forall \neg S. \neg C), \exists R.C, \forall \neg S. \neg C \}\]

\[
\mathcal{L}(z) = \{ C, \forall \neg R.(\forall \neg S. \neg C), \exists R.C \}\]

**Clash**
Non-finite Models

- With number restrictions some satisfiable concepts have only non-finite models
- E.g., testing $\neg C$ w.r.t. $T = \{ \top \subseteq \exists R.C, \top \subseteq \leq 1R^- \}$

$$\mathcal{L}(w) = \{ \neg C, \exists R.C, \leq 1R^- \}$$

$$\mathcal{L}(x) = \{ C, \exists R.C, \leq 1R^- \}$$

$$\mathcal{L}(y) = \{ C, \exists R.C, \leq 1R^- \}$$

model must be non-finite
Inadequacy of Dynamic Blocking

With non-finite models, even dynamic blocking not enough

E.g., testing $\neg C$ w.r.t. $\mathcal{T} = \{ \top \subseteq \exists R.(C \cap \exists R^-.\neg C), \top \subseteq \leq 1R^- \}$

\[
\begin{align*}
\mathcal{L}(w) &= \{ \neg C, \exists R.(C \cap \exists R^-.\neg C), \leq 1R^- \} \\
\mathcal{L}(x) &= \{(C \cap \exists R^-.\neg C), \exists R.(C \cap \exists R^-.\neg C), \leq 1R^-, C, \exists R^- . \neg C \} \\
\mathcal{L}(y) &= \{(C \cap \exists R^-.\neg C), \exists R.(C \cap \exists R^-.\neg C), \leq 1R^-, C, \exists R^- . \neg C \}
\end{align*}
\]

But $\exists R^- . \neg C \in \mathcal{L}(y)$ not satisfied

Inconsistency due to $\leq 1R^- \in \mathcal{L}(y)$ and $C \in \mathcal{L}(x)$
Double Blocking I

Problem due to $\exists R^-. \neg C$ term **only** satisfied in predecessor of blocking node

\[
\begin{align*}
\mathcal{L}(w) &= \{ \neg C, \exists R. (C \cap \exists R^- \neg C'), \leq 1R^- \} \\
R
\end{align*}
\]

\[
\begin{align*}
\mathcal{L}(x) &= \{ (C \cap \exists R^- \neg C'), \exists R. (C \cap \exists R^- \neg C'), \leq 1R^-, C, \exists R^- \neg C \} \\
x
\end{align*}
\]

Solution is **Double Blocking** (pairwise blocking)

- Predecessors of blocked and blocking nodes also considered
- In particular, $\exists R.C$ terms satisfied in predecessor of blocking node must also be satisfied in predecessor of blocked node $\neg C \in \mathcal{L}(w)$
Due to pairwise condition, block no longer holds.
Expansion continues and contradiction discovered.

\[
\begin{align*}
\mathcal{L}(w) &= \{ \neg C, \exists R. (C \cap \exists R^- \neg C), \leq 1 R^- \} \\
\mathcal{L}(x) &= \{ (C \cap \exists R^- \neg C), \exists R. (C \cap \exists R^- \neg C), \leq 1 R^-, C, \exists R^- \neg C, \neg C \} \\
\mathcal{L}(y) &= \{ (C \cap \exists R^- \neg C), \exists R. (C \cap \exists R^- \neg C), \leq 1 R^-, C, \exists R^- \neg C \}
\end{align*}
\]
## Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\mathcal{ALCN}$</strong></td>
<td>$\mathcal{ALCN}$</td>
<td>(wrt acyc. TBoxes)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **$\mathcal{I}$** inverse roles: h-child<sup>-</sup>
- **$\mathcal{N}$** NRs: ($\geq n$ h-child)
- **Q** Qual. NRs: ($\geq n$ h-child Blond)
- **O** nominals: "John" is a concept
- **$\mathcal{F}$** feature chain (dis)agreement
- **$\mathcal{r}^+$** declare roles as transitive
- **$\sqcap, \sqcup$** Boolean ops on roles
## Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{ALCUN}$ (NP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without $\exists$, only $\neg A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{ALCN}$ (wrt acyc. TBoxes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\mathcal{I}$ inverse roles: $\text{h-child}^{-}$

$\mathcal{N}$ NRs: $(\geq n \text{ h-child})$

$\mathcal{Q}$ Qual. NRs: $(\geq n \text{ h-child Blond})$

$\mathcal{O}$ nominals: "John" is a concept

$\mathcal{F}$ feature chain (dis)agreement

$\mathcal{R}^+$ declare roles as transitive

$\mathcal{\neg \land \lor}$ Boolean ops on roles

TU Dresden Germany

28
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALCN</td>
<td>(NP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>without $\exists$, only $\neg A$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **$\mathcal{ALN}$** without $\sqcap$

---

**Qualitative Networks (NRs):**

- $\mathcal{I}$ inverse roles: h-child$
- $\mathcal{N}$ NRs: $(\geq n$ h-child$
- $\mathcal{Q}$ Qualitative NRs: $(\geq n$ h-child blond$
- $\mathcal{O}$ Nominals: “John” is a concept$
- $\mathcal{F}$ Feature chain (dis)agreement$
- $\mathcal{P}$ Declare roles as transitive$
- $\neg \lor \land$ Boolean ops on roles$

---

**Language:**

- ALCN (wrt acyclic TBoxes)
Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC reg</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC NO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC u</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALCIQ</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>R+</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>QI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC N</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC N</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALCN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALCN</strong></td>
<td>(NP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC (NP)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ (NP)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ w</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ R+</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC+ QI R+ GCIs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inverse roles:** h-child

**NRs:** (≥ n h-child)

**Qual. NRs:** (≥ n h-child Blond)

**Nominals:** “John” is a concept

**Feature chain (dis)agreement**

**R+** declare roles as transitive

**Boolean ops on roles**

---

T U Dresden
Germany

30
# Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>reg</strong></td>
<td>add regular roles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALCNO</strong></td>
<td>add universal role</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>u</strong></td>
<td><strong>ALCIQ</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>+</strong></td>
<td>GCIs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Q</strong></td>
<td>still in ExpTime</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>wrt</strong></td>
<td>general TBoxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Q</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Qual.</strong></td>
<td>NRs: (≥ n h-child)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N</strong></td>
<td>NRs: (≥ n h-child)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>inverse roles: h-child</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>T</strong></td>
<td>feature chain (dis)agreement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>O</strong></td>
<td>nominals: &quot;John&quot; is a concept</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>R</strong></td>
<td>declare roles as transitive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>¬∪</strong>, <strong>¬∩</strong>, <strong>∪</strong></td>
<td>Boolean ops on roles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ALC** without \(\sqcup\)
subsumption of **FL\(\_\)\(0\)**
\(\Box\) and \(\forall\) only

**ALCIQO**

**ALC**
without \(\exists\), only \(\neg A\)

**ALCNO**
without \(\sqcup\) and NRs, only \(\neg A\)

**ALCIQ**

**ALCN**
(wrt acyclic TBoxes)
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALN</strong> (NP) without $\exists$, only $\neg A$</td>
<td><strong>ALC^N</strong> (wrt acyc. TBoxes)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ALC</strong> (co-NP) without $\sqcap$ and NRs, only $\neg A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **ALN** without $\sqcup$
- **FL** without $\exists$
- **FL** (co-NP)

- Subsumption of $\mathcal{FL}_0$
- $\Box$ and $\forall$ only

#### Qual. NRs: $(\geq n)$ h-child
- **ALC** without $\exists$, only $\neg A$ (co-NP)
- **ALC** (wrt acyc. TBoxes)

- Inverse roles: h-child
- Nominals: “John” is a concept
- Feature chain (dis)agreement
- Declare roles as transitive
- Boolean ops on roles
| Complexity of DLs: Overview of the Complexity of Concept Consistency |
|---|---|---|---|---|
| P | (co-)NP | PSpace | ExpTime | NExpTime |
| **LOCN** | (NP) | **ALCN** | (wrt acyc. TBoxes) |
| without $\exists$, only $\neg A$ | | |
| **ALE** | (co-NP) | |
| without $\sqcup$ and NRs, only $\neg A$ | | |
| subsumption of $\mathcal{FL}_0$ | subsumption of $\mathcal{FL}_0$ (co-NP) | wrt acyc. TBoxes |
| $\Box$ and $\forall$ only | |
| $I$ inverse roles: h-child$^-$ | |
| $\mathcal{N}$ NRs: $(\geq n$ h-child) | |
| $\mathcal{Q}$ Qual. NRs: $(\geq n$ h-child Blond) | |
| $\mathcal{O}$ nominals: "John" is a concept | |
| $\mathcal{F}$ feature chain (dis)agreement | |
| $f_i, g_i$ functional roles sensitive | |
| $f_1 \cdots f_n \downarrow g_1 \cdots g_m$ and $f_1 \cdots f_n \uparrow g_1 \cdots g_m$ | |
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{ALN}) without (\sqcap)</td>
<td>(\mathcal{ALCUN}) (NP) without (\exists), only (\neg A)</td>
<td>(\mathcal{ALCN}) (wrt acyclic TBoxes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\mathcal{ALC}) without (\sqcap) and NRs, only (\neg A)</td>
<td>(\mathcal{ALE}) (co-NP)</td>
<td>(\mathcal{ALCTI}_{Q,R^+})</td>
<td>(\mathcal{ALCHI}_{Q,R^+}) add role hierarchies</td>
<td></td>
</tr>
</tbody>
</table>

| \(\mathcal{ fus}\) inverse roles: h-child\(^{-}\) | \(\mathcal{N}\) NRs: \((\geq n)\) h-child | \(\mathcal{Q}\) Qual. NRs: \((\geq n)\) h-child Blond | \(\mathcal{O}\) nominals: "John" is a concept | \(\mathcal{F}\) feature chain (dis)agreement | \(\cdot_{R^+}\) declare roles as transitive |

\(\mathcal{ALCF}\) wrt acyclic TBoxes
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Complexity</th>
<th>Complexity</th>
<th>Complexity</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td><strong>(co-)NP</strong></td>
<td><strong>PSpace</strong></td>
<td><strong>ExpTime</strong></td>
<td><strong>NExpTime</strong></td>
</tr>
<tr>
<td><strong>ALC</strong></td>
<td><strong>ALCUN</strong></td>
<td><strong>ALCN</strong></td>
<td><strong>ALCreg</strong></td>
<td><strong>ALC</strong></td>
</tr>
<tr>
<td><strong>without □</strong></td>
<td><strong>(NP)</strong> without (\exists), only (\neg A)</td>
<td><strong>(wrt acyc. TBoxes)</strong></td>
<td>add regular roles</td>
<td>add universal role</td>
</tr>
<tr>
<td><strong>ALE</strong></td>
<td><strong>AC</strong></td>
<td><strong>ACIQ</strong></td>
<td><strong>ALC_IQ</strong></td>
<td><strong>ALCHIQ</strong></td>
</tr>
<tr>
<td><strong>(co-NP)</strong></td>
<td><strong>wrt acyc. TBoxes</strong></td>
<td><strong>wrt general TBoxes</strong></td>
<td>add role hierarchies</td>
<td>wrt acyc. TBoxes</td>
</tr>
<tr>
<td><strong>subsumption of</strong></td>
<td><strong>subsumption of</strong></td>
<td><strong>FL_0</strong> (co-NP)</td>
<td><strong>FL_0</strong> (co-NP)</td>
<td><strong>FL_0</strong> (co-NP)</td>
</tr>
<tr>
<td><strong>(\Box)</strong> and <strong>(\forall)</strong> only</td>
<td><strong>(\Box)</strong> and <strong>(\forall)</strong> only</td>
<td><strong>(\Box)</strong> and <strong>(\forall)</strong> only</td>
<td><strong>(\Box)</strong> and <strong>(\forall)</strong> only</td>
<td><strong>(\Box)</strong> and <strong>(\forall)</strong> only</td>
</tr>
</tbody>
</table>

**I** inverse roles: \(h\text{-}child^-\)

**N** NRs: \(\geq n \text{ h-child}\)

**Q** Qual. NRs: \(\geq n \text{ h-child Blond}\)

**O** nominals: "John" is a concept

**F** feature chain (dis)agreement

\(\cdot R^+\) declare roles as transitive

\(\sim\) Boolean ops on roles

\(\text{TU Dresden}\)

\(\text{Germany}\)
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALN</strong></td>
<td></td>
<td><strong>ALUN</strong> (NP)</td>
<td><strong>ALCN</strong></td>
<td><strong>ALC_reg</strong> add regular roles</td>
<td><strong>ALC_u</strong> add universal role</td>
</tr>
<tr>
<td>without ⊓</td>
<td></td>
<td>without ∃, only ¬A</td>
<td>(wrt acyc. TBoxes)</td>
<td></td>
<td><strong>ALCHIQ_R^+</strong> add role hierarchies</td>
</tr>
<tr>
<td><strong>ALE</strong></td>
<td></td>
<td><strong>ALC_u_o</strong> (co-NP)</td>
<td><strong>ALC_I_Q_R^+</strong></td>
<td></td>
<td>+ QI still in ExpTime</td>
</tr>
<tr>
<td>without ⊓ and NRs, only ¬A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subsumption of FL₀</td>
<td>subsumption of FL₀ (co-NP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊓ and ∀ only</td>
<td>only ¬A</td>
<td>wrt acyc. TBoxes</td>
<td>wrt general TBoxes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Inverse roles:** h-child⁻
**NRs:** (≥ n h-child)
**Qual. NRs:** (≥ n h-child Blond)
**Nominals:** "John" is a concept

**Boolean ops on roles:** ⊓, ∪, ⊓ and ∀ only
**Feature chain (dis)agreement**
**Declare roles as transitive**

**ALCF**
**ALCF** wrt acyc. TBoxes

### Notes
- **FL₀** (co-NP)
- **ALCN** (wrt acyc. TBoxes)
- **ALC\_reg** add regular roles
- **ALC\_u** add universal role
- **ALCHIQ\_R^+** add role hierarchies
- **QI** still in ExpTime

**TU Dresden Germany**

36
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td>(co-)NP</td>
</tr>
<tr>
<td><strong>PSPACE</strong></td>
<td></td>
</tr>
<tr>
<td><strong>EXP TIME</strong></td>
<td>(wrt acyc. TBoxes)</td>
</tr>
<tr>
<td><strong>NEXP TIME</strong></td>
<td></td>
</tr>
</tbody>
</table>

- **$\text{ALCN}$ (NP)**: without $\exists$, only $\neg A$
- **$\text{ALE}$ (co-NP)**: without $\sqcap$ and NRs, only $\neg A$
- **$\text{ALCNO}$**: subsumption of $\text{FL}_0$ (co-NP) wrt acyc. TBoxes
- **$\text{ALCO}$**: subsumption of $\text{FL}_0$ wrt acyc. TBoxes
- **$\text{ALCF}$**: inverse roles: h-child
- **$\text{ALN}$**: NRs: ($\geq n$ h-child)
- **$\text{ALCQO}$**: Qual. NRs: ($\geq n$ h-child Blond)
- **$\text{ALCIQ}$**: nominals: ”John” is a concept
- **$\text{ALCF}$**: feature chain (dis)agreement
- **$\text{ALE}$**: $\sqcap$ only
- **$\text{ALC}$**: declare roles as transitive
- **$\text{ALCHIQ}$**: wrt acyc. TBoxes
- **$\text{QI}$**: still in ExpTime

- **$\text{ALC_req}$**: add regular roles
- **$\text{ALC}_u$**: add universal role
- **$\text{ALC}$**: wrt general TBoxes
- **$\text{ALCIQO}$**: add role hierarchies

---

Inversion of roles: $h$-child
Nominals: "John" is a concept
Boolean ops on roles
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>𝐄LCN</td>
<td>(NP)</td>
<td>𝐄LCN</td>
<td>𝐄LC_reg</td>
<td>+ 𝑄I still in ExpTime</td>
</tr>
<tr>
<td></td>
<td>without (\exists), only (\neg A)</td>
<td></td>
<td>𝐄LC_u</td>
<td></td>
</tr>
<tr>
<td>𝐄LCI</td>
<td>(co-NP)</td>
<td>𝐄LCI_nO</td>
<td>𝐄LCI_QO_r^+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>without (\sqcup) and NRs, only (\neg A)</td>
<td></td>
<td>𝐄LCI_QO</td>
<td></td>
</tr>
<tr>
<td>𝐄LCX</td>
<td>(co-NP)</td>
<td>𝐄LCX</td>
<td>𝐄LCX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with (\sqcup) and (\forall) only</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **I**: inverse roles: h-child
- **N**: NRs: \((\geq n) h\text{-child}\)
- **Q**: Qual. NRs: \((\geq n) h\text{-child Blond}\)
- **Q**: Nominals: "John" is a concept
- **F**: Feature chain (dis)agreement
- \(\cdot_{R^+}\): declare roles as transitive
- \(\cdot_{\lor,\land}\): Boolean ops on roles

Subsumption of \(\mathcal{F}L_0\) with \(\sqcap\) and \(\forall\) only.

Subsumption of \(\mathcal{F}L_0\) (co-NP) wrt acyc. TBoxes.

\(\neg\land\cup\) Boolean ops on roles.
### Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\mathcal{ALN}$</strong> (NP) without $\exists$, only $\neg A$</td>
<td>$\mathcal{ALC}_N$ (wrt acyc. TBoxes)</td>
<td>$\mathcal{ALC}_{reg}$ add regular roles</td>
<td>$\mathcal{ALC}_{u}$ add universal role</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{ALC}$ (co-NP) without $\sqcap$ and NRs, only $\neg A$</td>
<td>$\mathcal{ALCNO}$</td>
<td>$\mathcal{ALC}_u$ wrt general TBoxes</td>
<td>+ $\mathcal{QI}$ still in ExpTime</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{ALCNO}$</td>
<td>$\mathcal{ALCIO}$</td>
<td>$\mathcal{ALC}_I$ add role hierarchies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{ALCIO}$</td>
<td>$\mathcal{ALCIQO}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{ALCF}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $\mathcal{I}$ inverse roles: $h$-child
- $\mathcal{N}$ NRs: $(\geq n$ h-child$)$
- $\mathcal{Q}$ Qual. NRs: $(\geq n$ h-child Blond$)$
- $\mathcal{O}$ nominals: ”John” is a concept
- $\mathcal{F}$ feature chain (dis)agreement
- $\cdot_{R^+}$ declare roles as transitive
- $\cdot_{\sqcap,\sqcup}$ Boolean ops on roles
## Complexity of DLs: Overview of the Complexity of Concept Consistency

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>(co-)NP</td>
<td>PSpace</td>
<td>ExpTime</td>
<td>NExpTime</td>
</tr>
<tr>
<td>( \mathcal{ALC} ) without ( \sqcap )</td>
<td>( \mathcal{ALC}_{\text{reg}} ) add regular roles</td>
<td>( \mathcal{ALC}_{\text{u}} ) add universal role</td>
<td>( \mathcal{ALC} ) wrt general TBoxes</td>
<td>( \mathcal{ALC}_{\text{reg}} ) add regular roles</td>
</tr>
<tr>
<td>( \mathcal{ALC} ) without ( \sqcap ) and ( \forall ) only</td>
<td>( \mathcal{ALCNO} ) add role hierarchies</td>
<td>( \mathcal{ALC^{\neg}} )</td>
<td>( \mathcal{ALC^{\neg,\neg,\cup}} ) wrt acyc. TBoxes</td>
<td></td>
</tr>
<tr>
<td>( \mathcal{ALC} ) without ( \exists ), only ( \neg A )</td>
<td>( \mathcal{ALCN} ) (wrt acyc. TBoxes)</td>
<td>( \mathcal{ALCIQ} )</td>
<td>( \mathcal{ALCIQ} )</td>
<td></td>
</tr>
<tr>
<td>subsumption of ( \mathcal{FL}_0 ) ( \sqcap ) and ( \forall ) only</td>
<td>subsumption of ( \mathcal{FL}_0 ) (co-NP) wrt acyc. TBoxes</td>
<td>subsumption of ( \mathcal{FL}_0 ) (co-NP) wrt acyc. TBoxes</td>
<td>subsumption of ( \mathcal{FL}_0 ) (co-NP) wrt acyc. TBoxes</td>
<td></td>
</tr>
</tbody>
</table>

**Symbols and Concepts**
- \( \mathcal{ALCN} \): (NP) without \( \exists \), only \( \neg A \)
- \( \mathcal{ALC} \): (co-NP) without \( \sqcap \) and \( \forall \) only
- \( \mathcal{FL}_0 \): inverse roles: h-child
- \( \mathcal{ALCIQ} \): NRs: (\( \geq n \) h-child)
- \( \mathcal{ALC^{\neg}} \): Qual. NRs: (\( \geq n \) h-child Blond)
- \( \mathcal{ALC^{\neg,\neg,\cup}} \): nominals: ”John” is a concept
- \( \mathcal{ALC^{\neg}} \): feature chain (dis)agreement
- \( \mathcal{ALC^{\neg,\neg,\cup}} \): \( \cdot_{R^+} \) declare roles as transitive
- \( \mathcal{ALC^{\neg,\neg,\cup}} \): \( \cdot_{\neg,\cap,\cup} \) Boolean ops on roles
- \( \mathcal{ALC^{\neg,\neg,\cup}} \): wrt acyc. TBoxes

TU Dresden
Germany
We left out a variety of complexity results for

- **concept consistency** of other DLs
  (e.g., those with “concrete domains”)

- **other standard inferences**
  (e.g., deciding consistency of ABoxes w.r.t. TBoxes)

- **“non-standard” inferences** such as
  - matching and unification of concepts
  - rewriting concepts
  - least common subsumer (of a set of concepts)
  - most specific concept (of an ABox individual)
Implementing DL Systems
Naive Implementations

Problems include:

- **Space usage**
  - Storage required for tableaux datastructures
  - Rarely a serious problem in practice

- **Time usage**
  - Search required due to non-deterministic expansion
  - *Serious* problem in practice
  - Mitigated by:
    - Careful *choice of algorithm*
    - Highly *optimised implementation*
Careful Choice of Algorithm

Transitive roles instead of transitive closure
- Deterministic expansion of $\exists R.C'$, even when $R \in \mathbb{R}_+$
- (Relatively) simple blocking conditions
- Cycles always represent (part of) cyclical models

Direct algorithm/implementation instead of encodings
- GCI axioms can be used to “encode” additional operators/axioms
- Powerful technique, particularly when used with FL closure
- Can encode cardinality constraints, inverse roles, range/domain,

\[ \rightarrow \text{E.g., } (\text{domain } R.C') \equiv \exists R. \top \sqsubseteq C \]
- (FL) encodings introduce (large numbers of) axioms
- **BUT** even simple domain encoding is disastrous with large numbers of roles
Highly Optimised Implementation

Optimisation performed at 2 levels

Computing **classification** (partial ordering) of concepts

- Objective is to minimise 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 KB
  - E.g., to select order in which to classify concepts

Computing **subsumption** between concepts

- Objective is to minimise cost of single subsumption tests
- Small number of hard tests can dominate classification time
- Recent DL research has addressed this problem (with considerable success)
Optimising Subsumption Testing

**Optimisation techniques** broadly fall into 2 categories

- **Pre-processing optimisations**
  - Aim is to *simplify KB* and facilitate subsumption testing
  - Largely algorithm independent
  - Particularly important when KB contains GCI axioms

- **Algorithmic optimisations**
  - Main aim is to *reduce search space* due to non-determinism
  - Integral part of implementation
  - But often generally applicable to search based algorithms
Pre-processing Optimisations

Useful techniques include

☞ Normalisation and simplification of concepts
  ● Refinement of technique first used in KRIS system
  ● Lexically normalise 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
  ● Normalisation can discover “obvious” (un)satisfiability
  ● Structural analysis can discover “obvious” subsumption
Normalisation and Simplification

- Normalise 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.:
  - $(D \sqcap C) \sqcap (A \sqcap D) \rightarrow A \sqcap C \sqcap D$
  - $\forall R.\top \rightarrow \top$
  - $\ldots \sqcap C \sqcap \ldots \sqcap \neg C \sqcap \ldots \rightarrow \bot$

- Lazily unfold concepts in tableaux algorithm
  - Use names/pointers to refer to complex concepts
  - Only add structure as required by progress of algorithm
  - Detect clashes between lexically equivalent concepts

```
\{\text{HappyFather}, \neg \text{HappyFather}\} \rightarrow \text{clash}
\{\forall \text{has-child.}(\text{Doctor} \sqcup \text{Lawyer}), \exists \text{has-child.}(\neg \text{Doctor} \sqcap \neg \text{Lawyer})\} \rightarrow \text{search}
```
Reasoning w.r.t. set 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 already $2^{100}$
- GALEN (medical terminology) KB contains hundreds 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 $CN \sqsupseteq D$, add $\neg D$ only to node labels containing $\neg CN$
- Can expand definitions lazily
  - Only add definitions after other local (propositional) expansion
  - Only add definitions one step at a time
Absorption II

Transform GCIs into primitive definitions, e.g.
- \( CN \cap C \subseteq D \rightarrow CN \subseteq D \cup \neg C \)
- \( CN \cup C \supseteq D \rightarrow CN \supseteq D \cap \neg C \)

Absorb into existing primitive definitions, e.g.
- \( CN \subseteq A, \ CN \subseteq D \cup \neg C \rightarrow CN \subseteq A \cap (D \cup \neg C) \)
- \( CN \supseteq A, \ CN \supseteq D \cap \neg C \rightarrow CN \supseteq A \cup (D \cap \neg C) \)

Use lazy expansion technique with primitive definitions
- Disjunctions only added to “relevant” node labels

Performance improvements often too large to measure
- At least **four orders of magnitude** with \textsc{Galen} KB
Algorithmic Optimisations

Useful techniques include

☞ Avoiding redundancy in search branches
  ● Davis-Putnam style semantic branching search
  ● Syntactic branching with no-good list

☞ Dependency directed backtracking
  ● Backjumping
  ● Dynamic backtracking

☞ Caching
  ● Cache partial models
  ● Cache satisfiability status (of labels)

☞ Heuristic ordering of propositional and modal expansion
  ● Min/maximise constrainedness (e.g., MOMS)
  ● Maximise backtracking (e.g., oldest first)
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 with **GALEn KB** again **too large to measure**
Backjumping

E.g., if $\exists R. \neg A \cap \forall R. (A \cap B) \cap (C_1 \cup D_1) \cap \ldots \cap (C_n \cup D_n) \subseteq \mathcal{L}(x)$
Caching

- Cache the satisfiability status of a node label
  - Identical node labels often recur during expansion
  - Avoid re-solv- ing problems by caching satisfiability status
    - When $\mathcal{L}(x)$ initialised, 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
  - $C \not\subseteq D$ 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
  - Can use same technique in sub-problems
Summary

Naive implementation results in effective non-termination

Problem is caused by non-deterministic expansion (search)

- GCIs lead to huge search space

Solution (partial) is

- Careful choice of logic/algorithm
- Avoid encodings
- Highly optimised implementation

Most important optimisations are

- Absorption
- Dependency directed backtracking (backjumping)
- Caching

Performance improvements can be very large

- E.g., more than four orders of magnitude
DL Resources

- The official DL homepage: http://dl.kr.org/
- The DL mailing list: dl@dl.kr.org
- Patrick Lambrix’s very useful DL site (including lots of interesting links):
  http://www.ida.liu.se/labs/iislab/people/patla/DL/index.html
- The annual DL workshop:
  Proceedings on-line available at:
  http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/
- The OIL homepage: http://www.ontoknowledge.org/oil/
- More about i-com: http://www.cs.man.ac.uk/~franconi/
- More about FaCT: http://www.cs.man.ac.uk/~horrocks/