Description Logics—Basics, Applications, and More

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 of and reasoning about

➠ **terminological knowledge**

➠ **configurations**

➠ **ontologies**

➠ **database schemata**

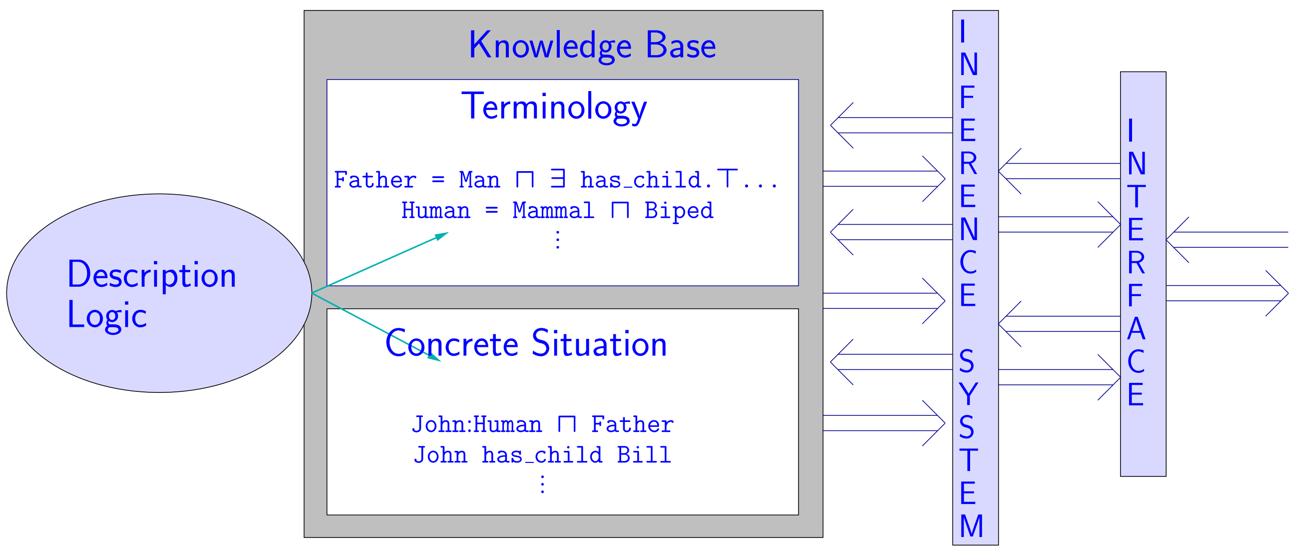
  – schema design, evolution, and query optimisation
  – source integration in heterogeneous databases/data warehouses
  – conceptual modelling of multidimensional aggregation

➠ . . .

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

- aka terminological KR systems, concept languages, etc.

Architecture of a Standard DL System



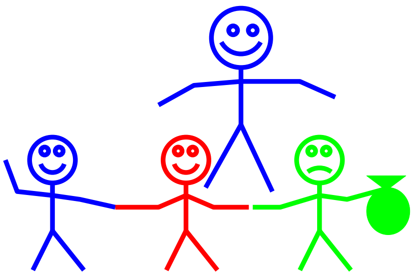
Introduction to DL I

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 $\mathcal{ALC}$



$\texttt{Man} \sqcap (\exists \texttt{has-child.Blue}) \sqcap$
$(\exists \texttt{has-child.Green}) \sqcap$
$(\forall \texttt{has-child.Happy} \sqcup \texttt{Rich})$

Introduction to DL: Syntax and Semantics of $\mathcal{ALC}$

Semantics given by means of an **interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$:

Constructor	Syntax	Example	Semantics
atomic concept	A	Human	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	R	likes	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
For C, D concepts and R a role name			
conjunction	$C \sqcap D$	Human $\sqcap$ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	Nice $\sqcup$ Rich	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\neg$ Meat	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
exists restrict.	$\exists R.C$	$\exists$has-child.Human	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restrict.	$\forall R.C$	$\forall$has-child.Blond	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

Introduction to DL: Other DL Constructors

Constructor	Syntax	Example	Semantics
number restriction	$(\geq n\ R)$	$(\geq 7$ has-child)	$\{x \mid \lvert\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}\rvert \geq n\}$
	$(\leq n\ R)$	$(\leq 1$ has-mother)	$\{x \mid \lvert\{y.\langle x,y\rangle \in R^{\mathcal{I}}\}\rvert \leq n\}$
inverse role	R^-	has-child$^-$	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$
trans. role	R^*	has-child*	$(R^{\mathcal{I}})^*$
concrete domain	$u_1,\ldots,u_n.P$	h-father·age, age. $>$	$\{x \mid \langle u_1^{\mathcal{I}},\ldots,u_n^{\mathcal{I}}\rangle \in P\}$
etc.			

Many different DLs/DL constructors have been investigated

Introduction to DL: Knowledge Bases: TBoxes

For terminological knowledge: **TBox** contains

Concept definitions $A \doteq C$ (A a concept name, C a complex concept)

Father $\doteq$ Man $\sqcap$ $\exists$has-child.Human

Human $\doteq$ Mammal $\sqcap$ $\forall$has-child$^-$.Human

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

Axioms $C_1 \sqsubseteq C_2$ (C_i complex concepts)

$\exists$favourite.Brewery $\sqsubseteq$ $\exists$drinks.Beer

$\leadsto$ restrict your models

An interpretation $\mathcal{I}$ **satisfies**

a **concept definition** $A \doteq C$ 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 $\mathcal{T}$

$\leadsto$ $\mathcal{I}$ is a **model** of $\mathcal{T}$

Introduction to DL: Knowledge Bases: ABoxes

For assertional knowledge: **ABox contains**

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

Role assertions $\langle a_1, a_2 \rangle : R$ (a_i individual names, R a role)
$\langle$John, Bill$\rangle$: has-child

An interpretation $\mathcal{I}$ **satisfies**

a **concept assertion** $a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$

a **role assertion** $\langle a_1, a_2 \rangle : R$ iff $\langle a_1^{\mathcal{I}}, a_2^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$

an **ABox** $\mathcal{A}$ iff $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$
$\rightsquigarrow$ $\mathcal{I}$ is a **model** of $\mathcal{A}$

Introduction to DL: Basic Inference Problems

Subsumption: $C \sqsubseteq D$ Is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all interpretations $\mathcal{I}$?

w.r.t. TBox $\mathcal{T}$: $C \sqsubseteq_{\mathcal{T}} D$ Is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{T}$?

$\rightsquigarrow$ structure your knowledge, compute taxonomy

Consistency: Is C consistent w.r.t. $\mathcal{T}$? Is there a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$?

of ABox $\mathcal{A}$: Is $\mathcal{A}$ consistent? Is there a model of $\mathcal{A}$?

of KB ($\mathcal{T}$,$\mathcal{A}$): Is ($\mathcal{T}$,$\mathcal{A}$) consistent? Is there a model of both $\mathcal{T}$ and $\mathcal{A}$?

Inference Problems are **closely related:**

$C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is **in**consistent w.r.t. $\mathcal{T}$,
(no model of $\mathcal{I}$ has an instance of $C \sqcap \neg D$)

C is consistent w.r.t. $\mathcal{T}$ iff **not** $C \sqsubseteq_{\mathcal{T}} A \sqcap \neg A$

$\rightsquigarrow$ **Decision Procdures for consistency (w.r.t. TBoxes) suffice**

Introduction to DL: Basic Inference Problems II

For most DLs, the basic inference problems are **decidable**,
with complexities between **P** and **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

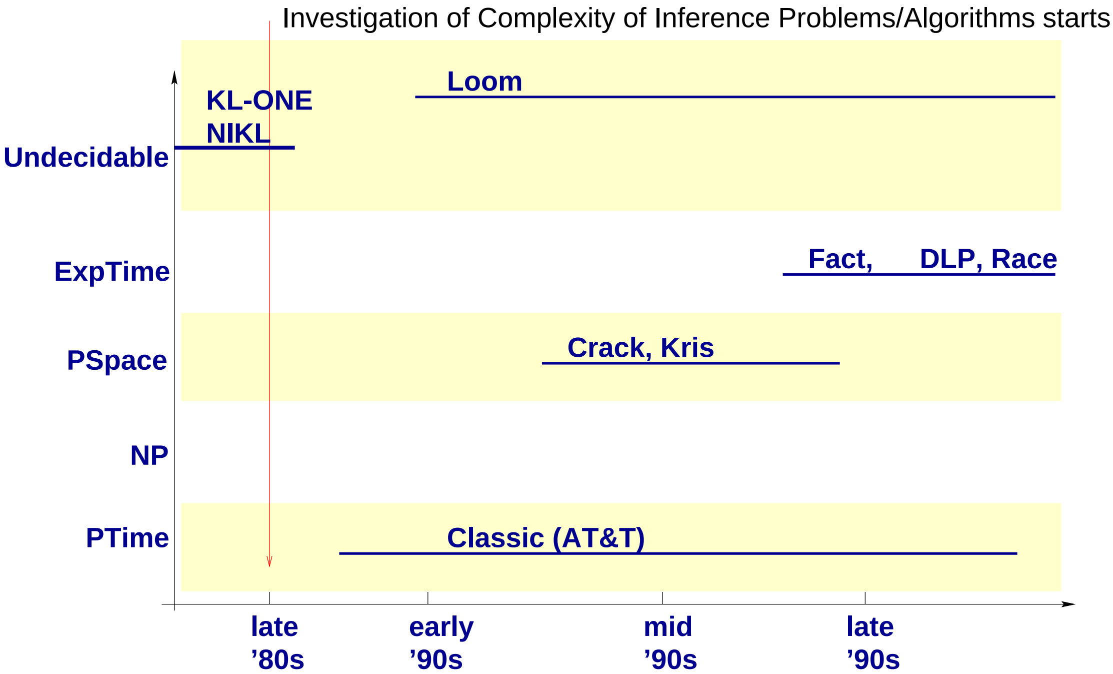
➠ consistency is semi-decidable, then subsumption is **not** semi-decidable

➠ Quest for a "highly expressive" DL with "practicable" inference problems

where **expressiveness** depends on the application
practicability changed over the time

Introduction to DL: History

Complexity of Inferences provided by DL systems over the time



Introduction to DL: State-of-the-implementation-art

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)

Relationship with Other Logical Formalisms: First Order Predicate Logic

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:

$$\begin{aligned} t_x(A) &= \mathrm{A}(x), & t_y(A) &= \mathrm{A}(y),\\ t_x(C \sqcap D) &= t_x(C) \wedge t_x(D), & t_y(C \sqcap D) &= t_y(C) \wedge t_y(D),\\ t_x(C \sqcup D) &= t_x(C) \vee t_x(D), & t_y(C \sqcup D) &= t_y(C) \vee t_y(D),\\ t_x(\exists R.C) &= \exists y.\mathrm{R}(x,y) \wedge t_y(C), & t_y(\exists R.C) &= \exists x.\mathrm{R}(y,x) \wedge t_x(C),\\ t_x(\forall R.C) &= \forall y.\mathrm{R}(x,y) \Rightarrow t_y(C), & t_y(\forall R.C) &= \forall x.\mathrm{R}(y,x) \Rightarrow t_x(C). \end{aligned}$$

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

$$\Phi_{\mathcal{T}} = \forall x. \bigwedge_{1 \le i \le n} t_x(C_i) \Leftrightarrow t_x(D_i)$$

Relationship with Other Logical Formalisms: First Order Predicate Logic II

$$C \text{ is consistent iff its translation } t_x(C) \text{ is satisfiable,}$$
$$C \text{ is consistent w.r.t. } \mathcal{T} \text{ iff its translation } t_x(C) \wedge \Phi_{\mathcal{T}} \text{ is satisfiable,}$$
$$C \sqsubseteq D \text{ iff } t_x(C) \Rightarrow t_x(D) \text{ is valid}$$
$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } \Phi_t \Rightarrow \forall x.(t_x(C) \Rightarrow t_x(D)) \text{ is valid.}$$

$\rightsquigarrow$ $\mathcal{ALC}$ 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

Relationship with Other Logical Formalisms: Modal Logics

DLs and Modal Logics are closely related:

$$\mathcal{ALC} \rightleftarrows \text{multi-modal } \mathbf{K}\text{:}$$

$$C \sqcap D \rightleftarrows C \wedge D, \qquad C \sqcup D \rightleftarrows C \vee D$$
$$\neg C \rightleftarrows \neg C \quad ,$$
$$\exists R.C \rightleftarrows \langle R \rangle C \quad , \qquad \forall R.C \rightleftarrows [R]C$$

transitive roles $\dot\rightleftarrows$ transitive frames (e.g., in $\mathbf{K4}$)

regular expressions on roles $\dot\rightleftarrows$ regular expressions on programs (e.g., in PDL)

inverse roles $\dot\rightleftarrows$ converse programs (e.g., in C-PDL)

number restrictions $\dot\rightleftarrows$ deterministic programs (e.g., in D-PDL)

➩ no TBoxes available in modal logics
  $\rightsquigarrow$ "internalise" axioms using a universal role u: $C \doteq D \rightleftarrows [u](C \Leftrightarrow D)$

➩ no ABox available in modal logics $\rightsquigarrow$ use nominals

Applications of Description Logics

Application Areas I

☞ 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** ($\mathcal{SHIQ}$)
- DL **reasoning** can be used to, e.g.,
  - ➙ Support ontology design and maintenance
  - ➙ Classify resources w.r.t. ontologies

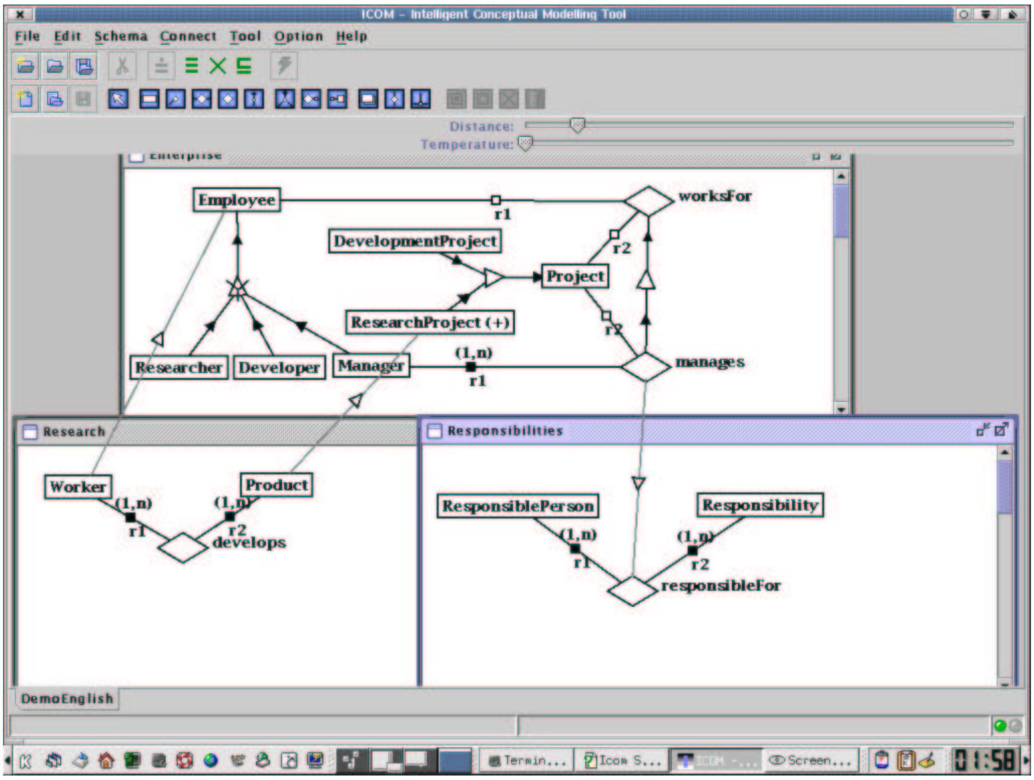
Application Areas II

- ☞ 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

- ☞ $\mathcal{DLR}$ (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- ☞ Satisfiability preserving mapping to $\mathcal{SHIQ}$ 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 $\mathcal{SHIQ}$
  - 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.

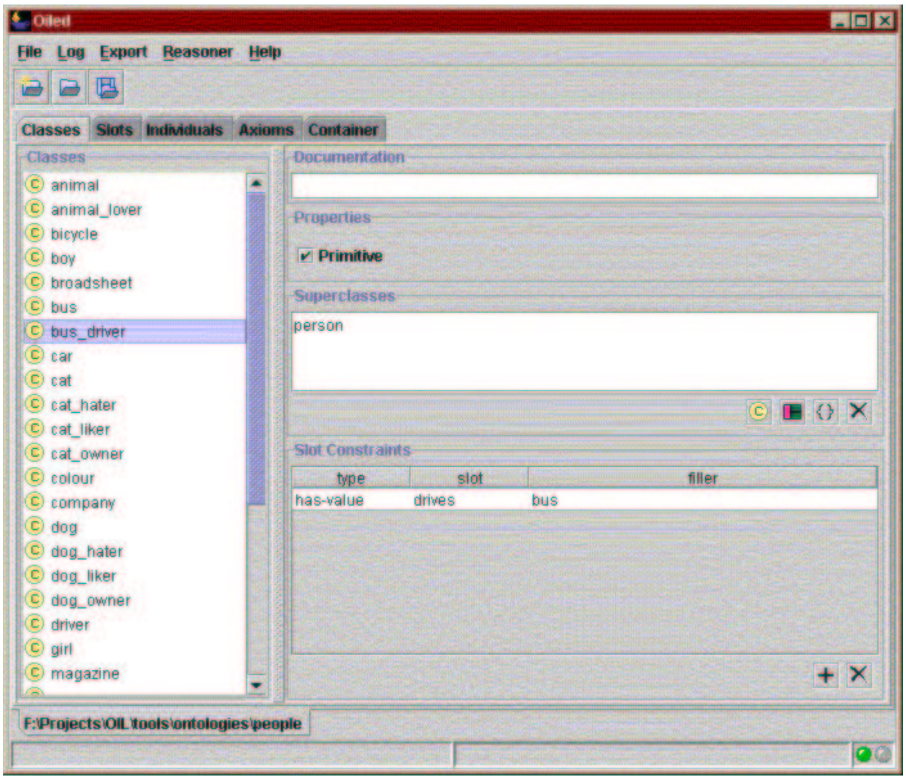
  Smoking $\sqsubseteq$ $\exists$causes.Cancer plus Cancer $\sqsubseteq$ $\exists$causes.Death
  $\Rightarrow$ Cancer $\sqsubseteq$ FatalThing
- GCIs represent additional non-definitional knowledge

  Stomach-Ulcer $\doteq$ Ulcer $\sqcap$ $\exists$hasLocation.Stomach plus
  Stomach-Ulcer $\sqsubseteq$ $\exists$hasLocation.Lining-Of-Stomach
  $\Rightarrow$ Ulcer $\sqcap$ $\exists$hasLocation.Stomach $\sqsubseteq$ OrganLiningLesion
- Inverse roles capture e.g. causes/causedBy relationship

  Death $\sqcap$ $\exists$causedBy.Smoking $\sqsubseteq$ PrematureDeath
  $\Rightarrow$ Smoking $\sqsubseteq$ CauseOfPrematureDeath
- Cardinality restrictions add consistency constraints

  BloodPressure $\sqsubseteq$ $\exists$hasValue.$($High $\sqcup$ Low$)$ $\sqcap$ $\leqslant 1$hasValue plus
  High $\sqsubseteq$ $\neg$Low $\Rightarrow$ HighLowBloodPressure $\sqsubseteq$ $\bot$

OilEd Demo



Reasoning Procedures: Deciding Consistency of $\mathcal{ALCN}$ Concepts

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., $\rightarrow_\sqcap$, $\rightarrow_\exists$)
- 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 $\Delta^{\mathcal{I}}$, labelled with sub-concepts of C_0
  edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- works on concepts in **negation normal form**: push negation inside using de Morgan' laws and

$$\neg(\exists R.C) \rightsquigarrow \forall R.\neg C \qquad \neg(\forall R.C) \rightsquigarrow \exists R.\neg C$$
$$\neg(\leq n\ R) \rightsquigarrow (\geq (n+1)R) \qquad \neg(\geq n\ R) \rightsquigarrow (\leq (n-1)R) \quad (n \geq 1)$$
$$\neg(\geq 0\ R) \rightsquigarrow A \sqcap \neg A$$

- 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,

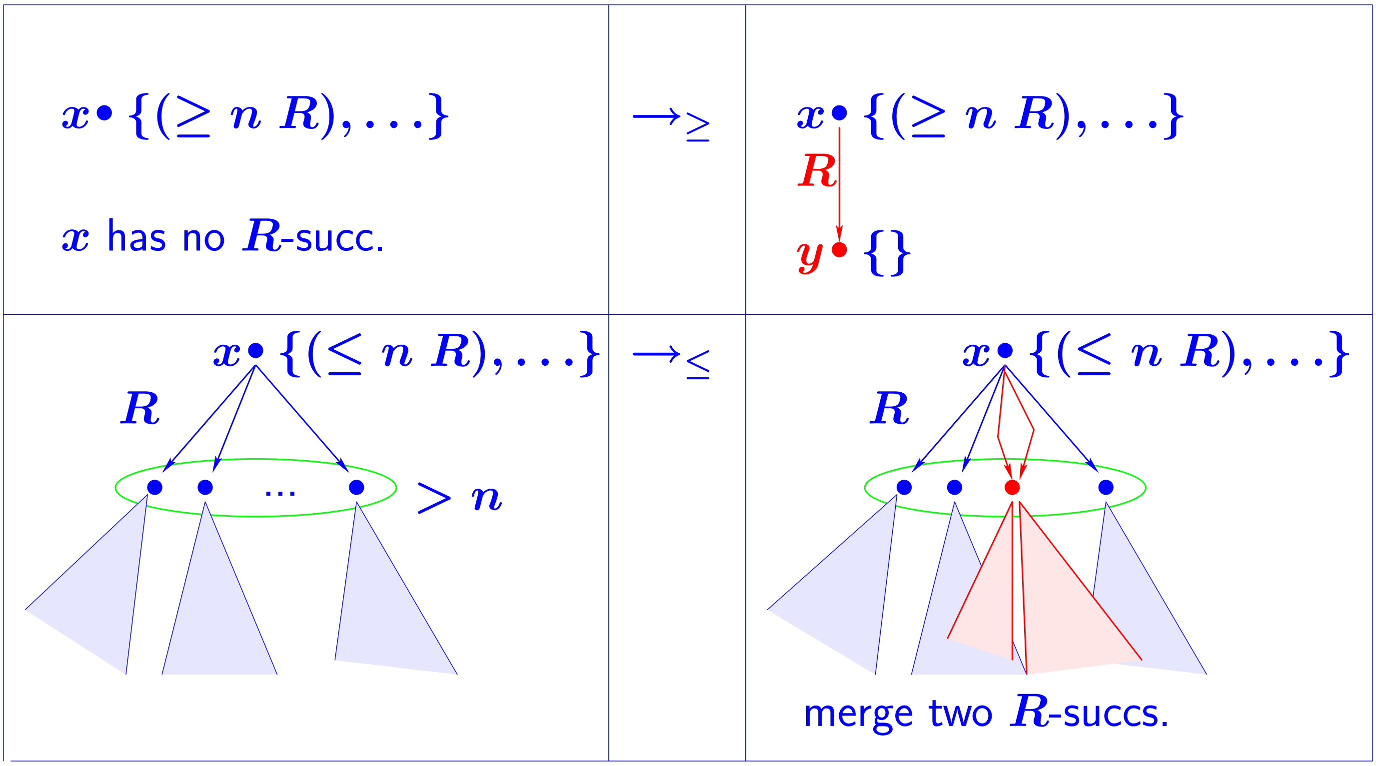
$$\{A, \neg A\} \subseteq \mathcal{L}(x) \text{ or}$$
$$\{(\geq m\ R), (\leq n\ R)\} \subseteq \mathcal{L}(x) \text{ for } n < m$$

- returns "C_0 is consistent" if rules can be applied s.t. they yield
  clah-free, complete (no more rules apply) tree

Reasoning Procedures: $\mathcal{ALC}$ Tableau Rules

$x\bullet\ \{C_1 \sqcap C_2, \ldots\}$	$\rightarrow_\sqcap$	$x\bullet\ \{C_1 \sqcap C_2, C_1, C_2, \ldots\}$
$x\bullet\ \{C_1 \sqcup C_2, \ldots\}$	$\rightarrow_\sqcup$	$x\bullet\ \{C_1 \sqcup C_2, C, \ldots\}$ for $C \in \{C_1, C_2\}$
$x\bullet\ \{\exists R.C, \ldots\}$	$\rightarrow_\exists$	$x\bullet\ \{\exists R.C, \ldots\}$ R ↓ $y\bullet\ \{C\}$
$x\bullet\ \{\forall R.C, \ldots\}$ R ↓ $y\bullet\ \{\ldots\}$	$\rightarrow_\forall$	$x\bullet\ \{\forall R.C, \ldots\}$ R ↓ $y\bullet\ \{\ldots, C\}$

Reasoning Procedures: $\mathcal{N}$ Tableau Rules



Reasoning Procedures: Soundness and Completeness

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

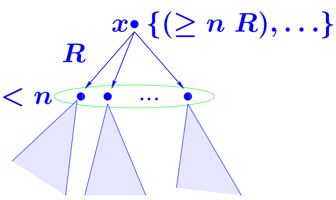
1. (Termination) The algorithm "monotonically" constructs a tree whose
   depth is linear in $|C_0|$: quantifier depth decreases from node to succs.
   breadth is linear in $|C_0|$ (even if number in NRs are coded binarily)

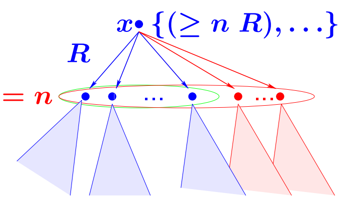
2. (Canonical model) Complete, clash-free tree $\mathbf{T}$ defines a (tree) pre-model $\mathcal{I}$:

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

   $\rightsquigarrow$ Easy to that $C \in \mathfrak{L}(x) \Rightarrow x \in C^{\mathcal{I}}$ — if $C \neq (\geq n\ R)$
   If $(\geq n\ R) \in \mathfrak{L}(x)$, then x might have **less than** n R-successors, but the $\rightarrow_{\geq}$-rule ensures that there is ≥ 1 R-successor...

copy some R-successors (including sub-trees) to obtain n R-successors:





$\rightsquigarrow$ **canonical tree model for input concept**

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

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

This easily implies clash-freenes of the tree generated.

Make the Tableau Algorithm run in PSpace:

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

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

$\rightsquigarrow$ tableau algorithm runs in NPspace (Savitch: NPspace = PSpace)

Reasoning Procedures: Extensibility

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** `has-child`$^-$

✔ $\mathcal{ALC}$ with **role conjunction**
$\exists(R \sqcap S).C$ and $\forall(R \sqcap S).C$

✔ **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

Reasoning Procedures: Extensibility II

Language extensions that require more elaborate techniques include

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

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

Reasoning Procedures II

Non-Termination

- As already mentioned, for $\mathcal{ALC}$ with **general axioms** basic algorithm is **non-terminating**
- **E.g.** if human $\sqsubseteq \exists$has-mother.human $\in \mathcal{T}$, then $\neg$human $\sqcup \exists$has-mother.human added to every node

w $\mathcal{L}(w) = \{$human, $(\neg$human $\sqcup \exists$has-mother.human$)$, $\exists$has-mother.human$\}$

has-mother

x $\mathcal{L}(x) = \{$human, $(\neg$human $\sqcup \exists$has-mother.human$)$, $\exists$has-mother.human$\}$

has-mother

y $\mathcal{L}(y) = \{$human, $(\neg$human $\sqcup \exists$has-mother.human$)$, $\exists$has-mother.human$\}$

Blocking

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

w $\mathcal{L}(w) = \{$human, $(\neg$human $\sqcup \exists$has-mother.human$)$, $\exists$has-mother.human$\}$

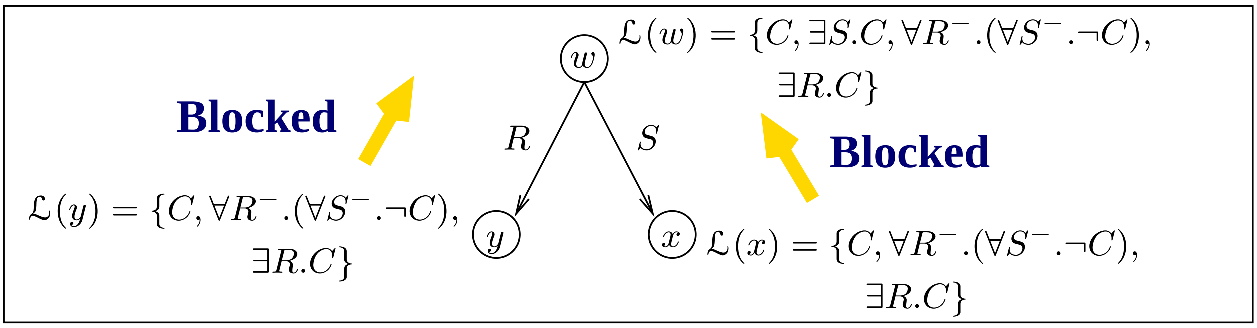
has-mother **Blocked**

x $\mathcal{L}(x) = \{$human, $(\neg$human $\sqcup \exists$has-mother.human$)\}$

Blocking with More Expressive DLs

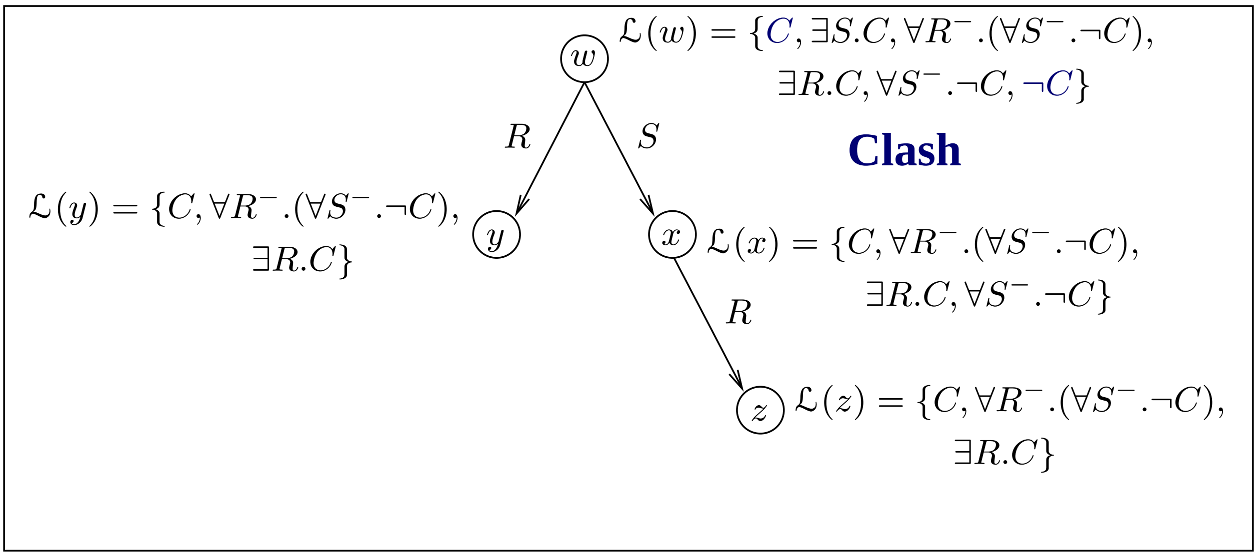
- Simple subset blocking may not work with more complex logics
- E.g., reasoning with inverse roles
  - Expanding node label can affect predecessor
  - Label of blocking node can affect predecessor
  - E.g., testing $C \sqcap \exists S.C$ w.r.t. Tbox

$$\mathcal{T} = \{\top \sqsubseteq \forall R^-.(\forall S^-.\neg C), \top \sqsubseteq \exists R.C\}$$



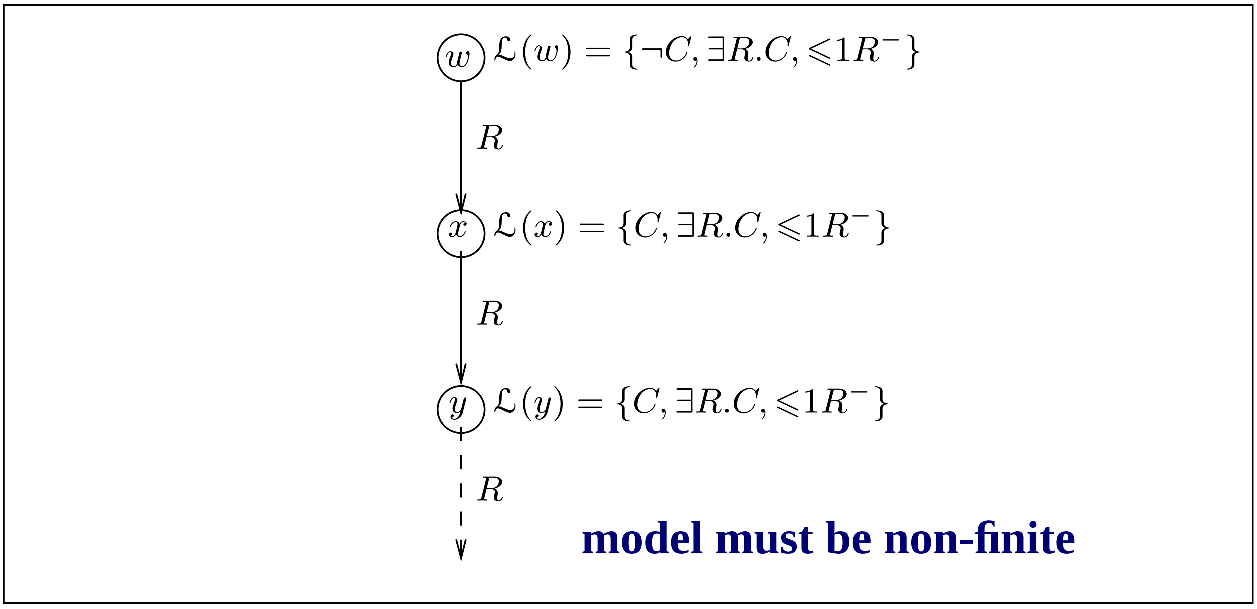
Dynamic Blocking

- 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



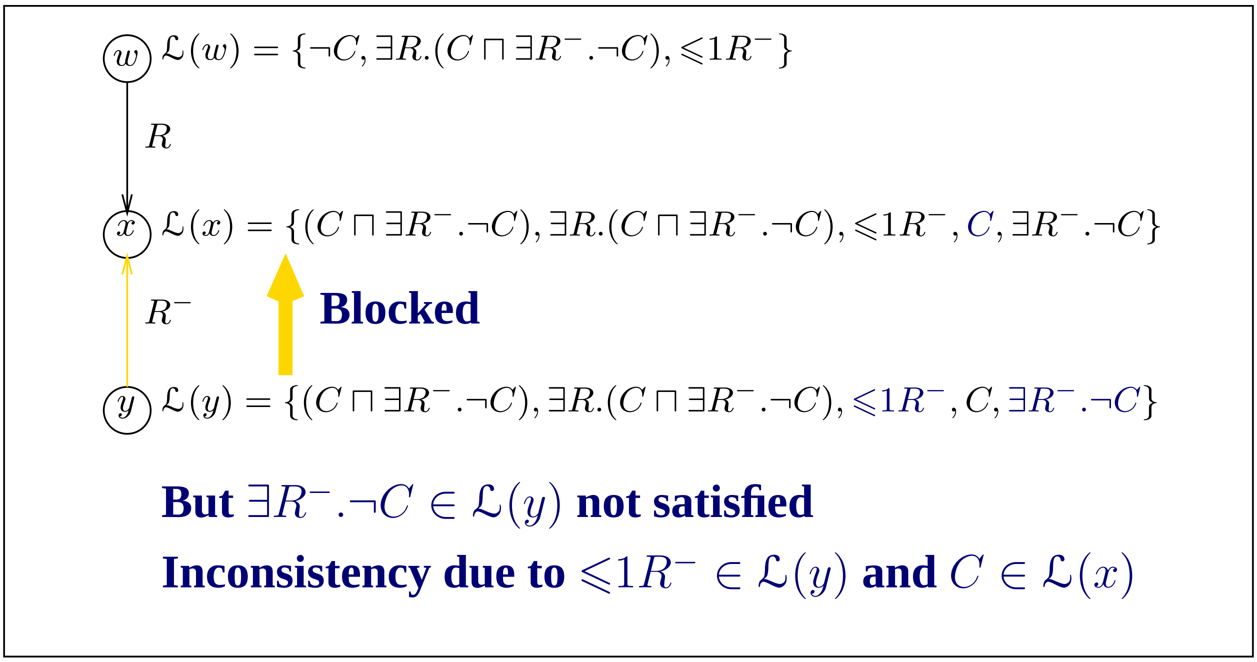
Non-finite Models

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



Inadequacy of Dynamic Blocking

- With non-finite models, even dynamic blocking not enough
- E.g., testing $\neg C$ w.r.t. $\mathcal{T} = \{\top \sqsubseteq \exists R.(C \sqcap \exists R^-.\neg C), \top \sqsubseteq \leqslant 1 R^-\}$



Double Blocking I

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

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

$w \xrightarrow{R} x$

$$x: \mathcal{L}(x) = \{(C \sqcap \exists R^{-}.\neg C), \exists R.(C \sqcap \exists R^{-}.\neg C), \leqslant 1 R^{-}, C, \exists R^{-}.\neg C\}$$

- 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 $\neg C \in \mathcal{L}(w)$

Double Blocking II

- Due to pairwise condition, block no longer holds
- Expansion continues and contradiction discovered

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

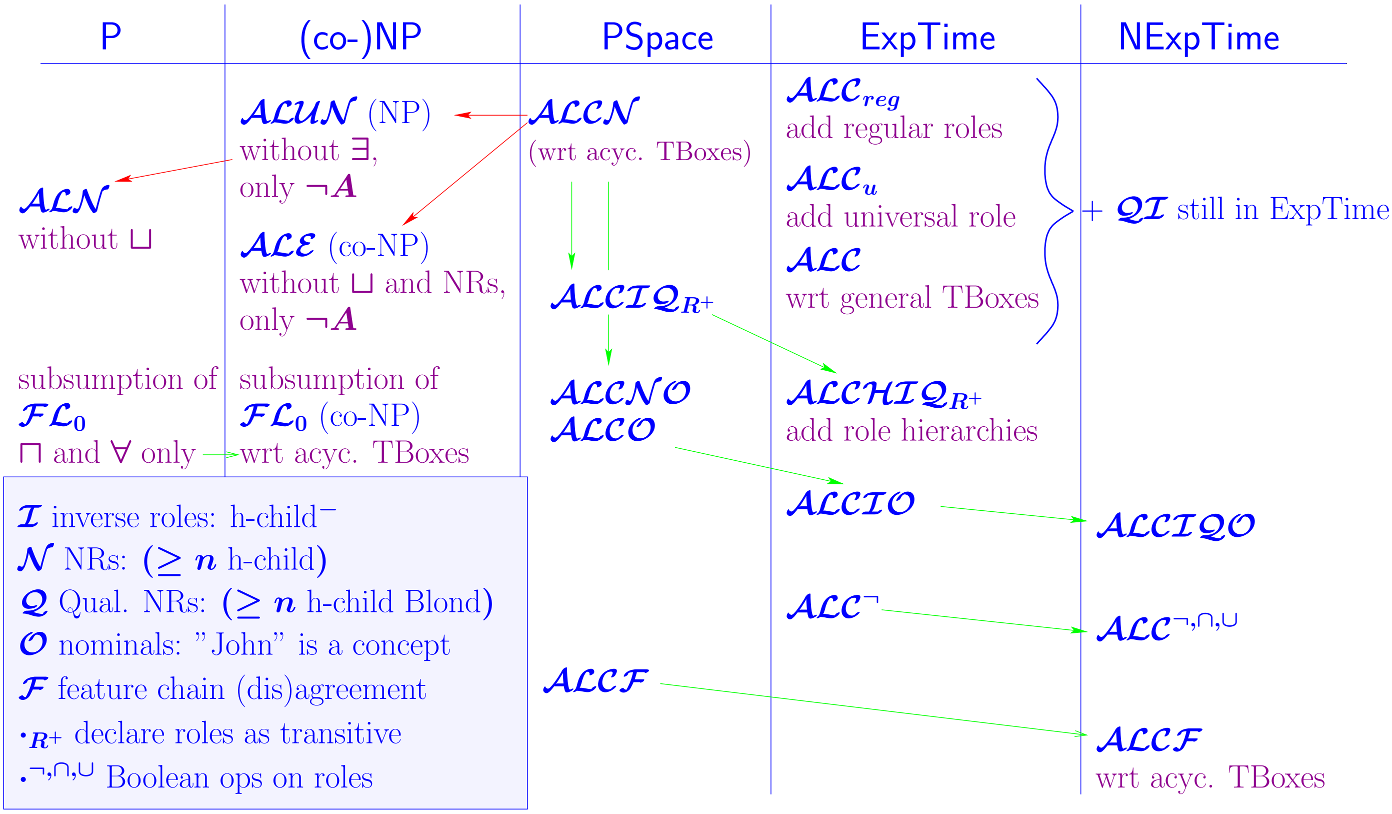
$w \xrightarrow{R} x$

$$x: \mathcal{L}(x) = \{(C \sqcap \exists R^{-}.\neg C), \exists R.(C \sqcap \exists R^{-}.\neg C), \leqslant 1 R^{-}, C, \exists R^{-}.\neg C, \neg C\}$$

$y \xrightarrow{R} x$ **Clash**

$$y: \mathcal{L}(y) = \{(C \sqcap \exists R^{-}.\neg C), \exists R.(C \sqcap \exists R^{-}.\neg C), \leqslant 1 R^{-}, C, \exists R^{-}.\neg C\}$$

Complexity of DLs: Overview of the Complexity of Concept Consistency



Complexity of DLs: What was left out

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 $R \in \mathbf{R}_+$
  - (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, ...
    - 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 $\mathcal{KRIS}$ 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.:
  - $\exists R.C \longrightarrow \neg\forall R.\neg C$
  - $C \sqcup D \longrightarrow \neg(\neg C \sqcap \neg D)$
- Simplify concepts, e.g.:
  - $(D \sqcap C) \sqcap (A \sqcap D) \longrightarrow A \sqcap C \sqcap D$
  - $\forall R.\top \longrightarrow \top$
  - $\ldots \sqcap C \sqcap \ldots \sqcap \neg C \sqcap \ldots \longrightarrow \bot$
- 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

$\{$HappyFather$, \neg$HappyFather$\} \longrightarrow$ **clash**

$\{\forall$has-child.$($Doctor $\sqcup$ Lawyer$), \exists$has-child.$(\neg$Doctor $\sqcap \neg$Lawyer$)\} \longrightarrow$ **search**

Absorption I

- 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 $\sqcap C \sqsubseteq D \longrightarrow$ CN $\sqsubseteq D \sqcup \neg C$
  - CN $\sqcup C \sqsupseteq D \longrightarrow$ CN $\sqsupseteq D \sqcap \neg C$
- Absorb into existing primitive definitions, e.g.
  - CN $\sqsubseteq A$, CN $\sqsubseteq D \sqcup \neg C \longrightarrow$ CN $\sqsubseteq A \sqcap (D \sqcup \neg C)$
  - CN $\sqsupseteq A$, CN $\sqsupseteq D \sqcap \neg C \longrightarrow$ CN $\sqsupseteq A \sqcup (D \sqcap \neg C)$
- 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 GALEN KB

Algorithmic Optimisations

Useful techniques include

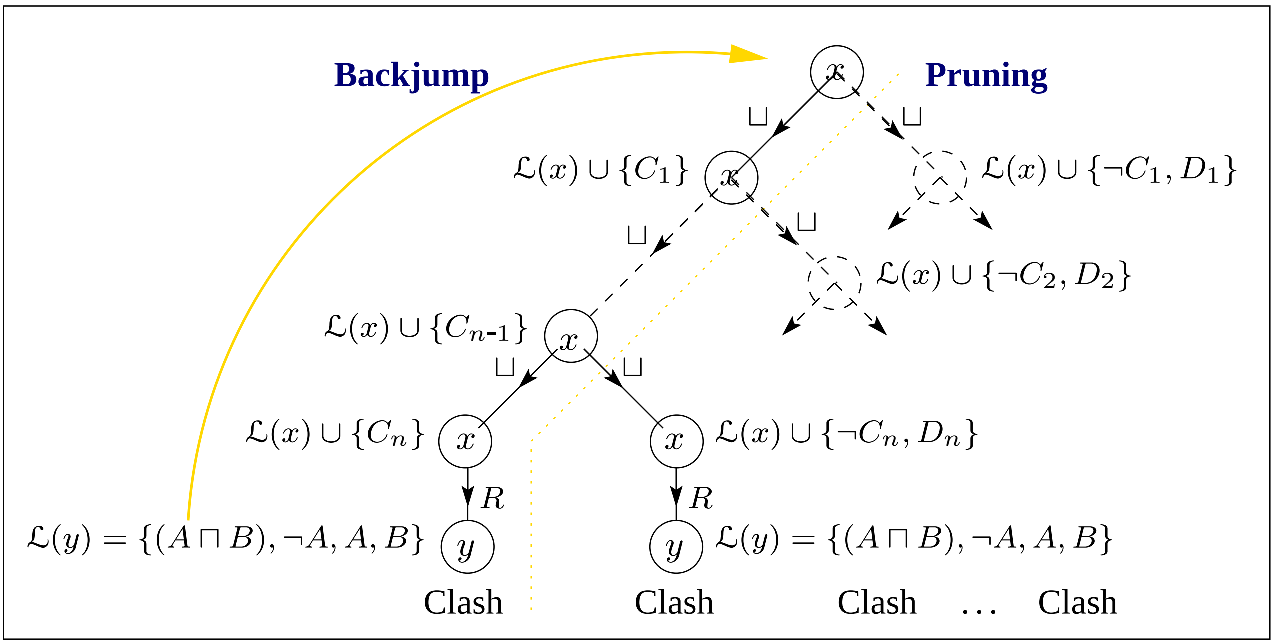
- Avoiding redundancy in search branches
  - Davis-Putnam style semantic branching search
  - Syntactic branching with no-good list
- Dependency directed backtracking
  - Backjumping
  - Dynamic backtracking
- Caching
  - Cache partial models
  - Cache satisfiability status (of labels)
- Heuristic ordering 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 $\exists R.\neg A \sqcap \forall R.(A \sqcap B) \sqcap (C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n) \subseteq \mathcal{L}(x)$



Caching

- Cache the satisfiability status of a node label
  - Identical node labels often recur during expansion
  - Avoid re-solving problems by caching satisfiability status
    - When $\mathcal{L}(x)$ 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
  - $C \not\sqsubseteq D$ if $C \sqcap \neg D$ is satisfiable
  - $C \sqcap \neg D$ satisfiable if models of C and $\neg D$ can be merged
  - If not, continue with standard subsumption test
  - 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:**

  DL2002 (co-located KR2002): `http://www.cs.man.ac.uk/dl2002`

  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/`