Description Logic: Axioms and Rules

Ian Horrocks

horrocks@cs.man.ac.uk

University of Manchester

Manchester, UK

Talk Outline

Talk Outline

Motivation: **The Semantic Web and DAML+OIL**

Talk Outline

- Motivation: **The Semantic Web and DAML+OIL**
- **Description Logics and Reasoning**
  - **Reasoning techniques**
  - **Implementing DL systems**

Talk Outline

- Motivation: **The Semantic Web and DAML+OIL**
- **Description Logics and Reasoning**
  - **Reasoning techniques**
  - **Implementing DL systems**
- **Axioms and Rules**

Talk Outline

- Motivation: **The Semantic Web and DAML+OIL**
- **Description Logics and Reasoning**
  - **Reasoning techniques**
  - **Implementing DL systems**
- **Axioms and Rules**
- **Research Challenges**

Talk Outline

- Motivation: **The Semantic Web and DAML+OIL**
- **Description Logics and Reasoning**
  - **Reasoning techniques**
  - **Implementing DL systems**
- **Axioms and Rules**
- **Research Challenges**
- **Summary**

The Semantic Web and DAML+OIL

Semantic Web Ontology Languages

US **DAML** programme (in cooperation with W3C and a cast of thousands) aim to develop so-called **Semantic Web**

Semantic Web Ontology Languages

US **DAML** programme (in cooperation with W3C and a cast of thousands) aim to develop so-called **Semantic Web**

☞ Most existing Web resources only human understandable

- Markup (HTML) provides **rendering information**
- Textual/graphical information for **human consumption**

Semantic Web Ontology Languages

US **DAML** programme (in cooperation with W3C and a cast of thousands) aim to develop so-called **Semantic Web**

- ☞ Most existing Web resources only human understandable
  - Markup (HTML) provides **rendering information**
  - Textual/graphical information for **human consumption**
- ☞ Semantic Web aims at machine understandability
  - **Semantic** markup will be added to web resources
  - Markup will use **Ontologies** for shared understanding

Semantic Web Ontology Languages

US **DAML** programme (in cooperation with W3C and a cast of thousands) aim to develop so-called **Semantic Web**

- Most existing Web resources only human understandable
  - Markup (HTML) provides **rendering information**
  - Textual/graphical information for **human consumption**
- Semantic Web aims at machine understandability
  - **Semantic** markup will be added to web resources
  - Markup will use **Ontologies** for shared understanding
- Requirement for a suitable ontology language
  - Compatible with existing Web standards (XML, RDF)
  - Captures common KR idioms
  - Formally specified and of “adequate expressive power”
  - Can provide reasoning support

Semantic Web Ontology Languages

US **DAML** programme (in cooperation with W3C and a cast of thousands) aim to develop so-called **Semantic Web**

- Most existing Web resources only human understandable
  - Markup (HTML) provides **rendering information**
  - Textual/graphical information for **human consumption**
- Semantic Web aims at machine understandability
  - **Semantic** markup will be added to web resources
  - Markup will use **Ontologies** for shared understanding
- Requirement for a suitable ontology language
  - Compatible with existing Web standards (XML, RDF)
  - Captures common KR idioms
  - Formally specified and of “adequate expressive power”
  - Can provide reasoning support
- DAML-ONT language developed to meet these requirements

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe…

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe...

- ☞ **OIL** language had been developed to meet similar requirements

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe…

☞ **OIL** language had been developed to meet similar requirements

- Extends existing Web standards (XML, RDF)
- Intuitive (frame) syntax plus high expressive power
- Well defined semantics via mapping to $\mathcal{SHIQ}$ DL
- Can use DL systems to reason with OIL ontologies

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe...

☞ **OIL** language had been developed to meet similar requirements

- Extends existing Web standards (XML, RDF)
- Intuitive (frame) syntax plus high expressive power
- Well defined semantics via mapping to $\mathcal{SHIQ}$ DL
- Can use DL systems to reason with OIL ontologies

☞ Two efforts merged to produce single language, **DAML+OIL**

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe…

- **OIL** language had been developed to meet similar requirements
  - Extends existing Web standards (XML, RDF)
  - Intuitive (frame) syntax plus high expressive power
  - Well defined semantics via mapping to $\mathcal{SHIQ}$ DL
  - Can use DL systems to reason with OIL ontologies
- Two efforts merged to produce single language, **DAML+OIL**
- Detailed specification agreed by **Joint EU/US Committee on Agent Markup Languages**

OIL and DAML+OIL

Meanwhile, somewhere in darkest Europe...

- **OIL** language had been developed to meet similar requirements
  - Extends existing Web standards (XML, RDF)
  - Intuitive (frame) syntax plus high expressive power
  - Well defined semantics via mapping to $\mathcal{SHIQ}$ DL
  - Can use DL systems to reason with OIL ontologies
- Two efforts merged to produce single language, **DAML+OIL**
- Detailed specification agreed by **Joint EU/US Committee on Agent Markup Languages**
- W3C Ontology Language WG has taken DAML+OIL as starting point

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

- ☞ Describes **structure** of the domain (i.e., a Tbox)
  - RDF used to describe specific **instances** (i.e., an Abox)

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

- ☞ Describes **structure** of the domain (i.e., a Tbox)
  - RDF used to describe specific **instances** (i.e., an Abox)
- ☞ Structure described in terms of **classes** (concepts) and **properties** (roles)

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

- ☞ Describes **structure** of the domain (i.e., a Tbox)
  - RDF used to describe specific **instances** (i.e., an Abox)
- ☞ Structure described in terms of **classes** (concepts) and **properties** (roles)
- ☞ Ontology consists of set of **axioms**
  - E.g., asserting class subsumption/equivalence

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

- Describes **structure** of the domain (i.e., a Tbox)
  - RDF used to describe specific **instances** (i.e., an Abox)
- Structure described in terms of **classes** (concepts) and **properties** (roles)
- Ontology consists of set of **axioms**
  - E.g., asserting class subsumption/equivalence
- Classes can be names or **expressions**
  - Various **constructors** provided for building class expressions

DAML+OIL Language Overview

DAML+OIL is an **ontology** language

- ☞ Describes **structure** of the domain (i.e., a Tbox)
  - RDF used to describe specific **instances** (i.e., an Abox)
- ☞ Structure described in terms of **classes** (concepts) and **properties** (roles)
- ☞ Ontology consists of set of **axioms**
  - E.g., asserting class subsumption/equivalence
- ☞ Classes can be names or **expressions**
  - Various **constructors** provided for building class expressions
- ☞ **Expressive power** determined by
  - Kinds of axiom supported
  - Kinds of class (and property) constructor supported

DAML+OIL

DAML+OIL

- ☞ Is a **Description Logic**

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)
- More precisely, DAML+OIL is $\mathcal{SHIQ}$

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)
- More precisely, DAML+OIL is $\mathcal{SHIQ}$
  - Plus **nominals**

DAML+OIL

- ☞ Is a **Description Logic** (but don't tell anyone)
- ☞ More precisely, DAML+OIL is $\mathcal{SHIQ}$
  - Plus **nominals**
  - Plus **datatypes** (simple concrete domains)

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)
- More precisely, DAML+OIL is $\mathcal{SHIQ}$
  - Plus **nominals**
  - Plus **datatypes** (simple concrete domains)
  - With RDFS based syntax

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)
- More precisely, DAML+OIL is $\mathcal{SHIQ}$
  - Plus **nominals**
  - Plus **datatypes** (simple concrete domains)
  - With RDFS based syntax
- $\mathcal{SHIQ}$/DAML+OIL was not built in a day (or even a year)
  - $\mathcal{SHIQ}$ is based on 15+ years of DL research

DAML+OIL

- Is a **Description Logic** (but don't tell anyone)
- More precisely, DAML+OIL is $\mathcal{SHIQ}$
  - Plus **nominals**
  - Plus **datatypes** (simple concrete domains)
  - With RDFS based syntax
- $\mathcal{SHIQ}$/DAML+OIL was not built in a day (or even a year)
  - $\mathcal{SHIQ}$ is based on 15+ years of DL research
- Can use DL reasoning with DAML+OIL
  - Existing $\mathcal{SHIQ}$ implementations support (most of) DAML+OIL

Why Reasoning Services?

Reasoning is important for:

Why Reasoning Services?

Reasoning is important for:

☞ Ontology **design**

- Check class consistency and (unexpected) implied relationships
- Particularly important with large ontologies/multiple authors

Why Reasoning Services?

Reasoning is important for:

- Ontology **design**
  - Check class consistency and (unexpected) implied relationships
  - Particularly important with large ontologies/multiple authors
- Ontology **integration**
  - Assert inter-ontology relationships
  - Reasoner computes integrated class hierarchy/consistency

Why Reasoning Services?

Reasoning is important for:

☞ Ontology **design**

- Check class consistency and (unexpected) implied relationships
- Particularly important with large ontologies/multiple authors

☞ Ontology **integration**

- Assert inter-ontology relationships
- Reasoner computes integrated class hierarchy/consistency

☞ Ontology **deployment**

- Determine if set of facts are consistent w.r.t. ontology
- Answer queries w.r.t. ontology, e.g., DQL

Why Decidable Reasoning?

Set of operators/axioms restricted so that reasoning is **decidable**

Why Decidable Reasoning?

Set of operators/axioms restricted so that reasoning is **decidable**

- ☞ Consistent with Semantic Web's **layered architecture**

Why Decidable Reasoning?

Set of operators/axioms restricted so that reasoning is **decidable**

- ☞ Consistent with Semantic Web's **layered architecture**
  - XML provides syntax transport layer
  - RDF provides basic relational language
  - RDFS provides basic ontological primitives
  - DAML+OIL provides (decidable) logical layer
  - Further layers (e.g., **rules**) will extend DAML+OIL
    - ➜ Extensions will almost certainly be **undecidable**

Why Decidable Reasoning?

Set of operators/axioms restricted so that reasoning is **decidable**

- ☞ Consistent with Semantic Web's **layered architecture**
  - XML provides syntax transport layer
  - RDF provides basic relational language
  - RDFS provides basic ontological primitives
  - DAML+OIL provides (decidable) logical layer
  - Further layers (e.g., **rules**) will extend DAML+OIL
    - ➜ Extensions will almost certainly be **undecidable**
- ☞ Facilitates provision of **reasoning services**

Why Decidable Reasoning?

Set of operators/axioms restricted so that reasoning is **decidable**

- ☞ Consistent with Semantic Web's **layered architecture**
  - XML provides syntax transport layer
  - RDF provides basic relational language
  - RDFS provides basic ontological primitives
  - DAML+OIL provides (decidable) logical layer
  - Further layers (e.g., **rules**) will extend DAML+OIL
    - ➜ Extensions will almost certainly be **undecidable**
- ☞ Facilitates provision of **reasoning services**
  - Known algorithms
  - Implemented systems
  - Evidence of **empirical tractability** (for ontology reasoning)

Reasoning Support for Ontology Design: OilEd

OilEd is a DAML+OIL **ontology editor** with DL reasoning support

Reasoning Support for Ontology Design: OilEd

OilEd is a DAML+OIL **ontology editor** with DL reasoning support

- ☞ **Frame based** interface (inspired by Protégé)
  - Classes defined by superclass(es) plus slot constraints

Reasoning Support for Ontology Design: OilEd

OilEd is a DAML+OIL **ontology editor** with DL reasoning support

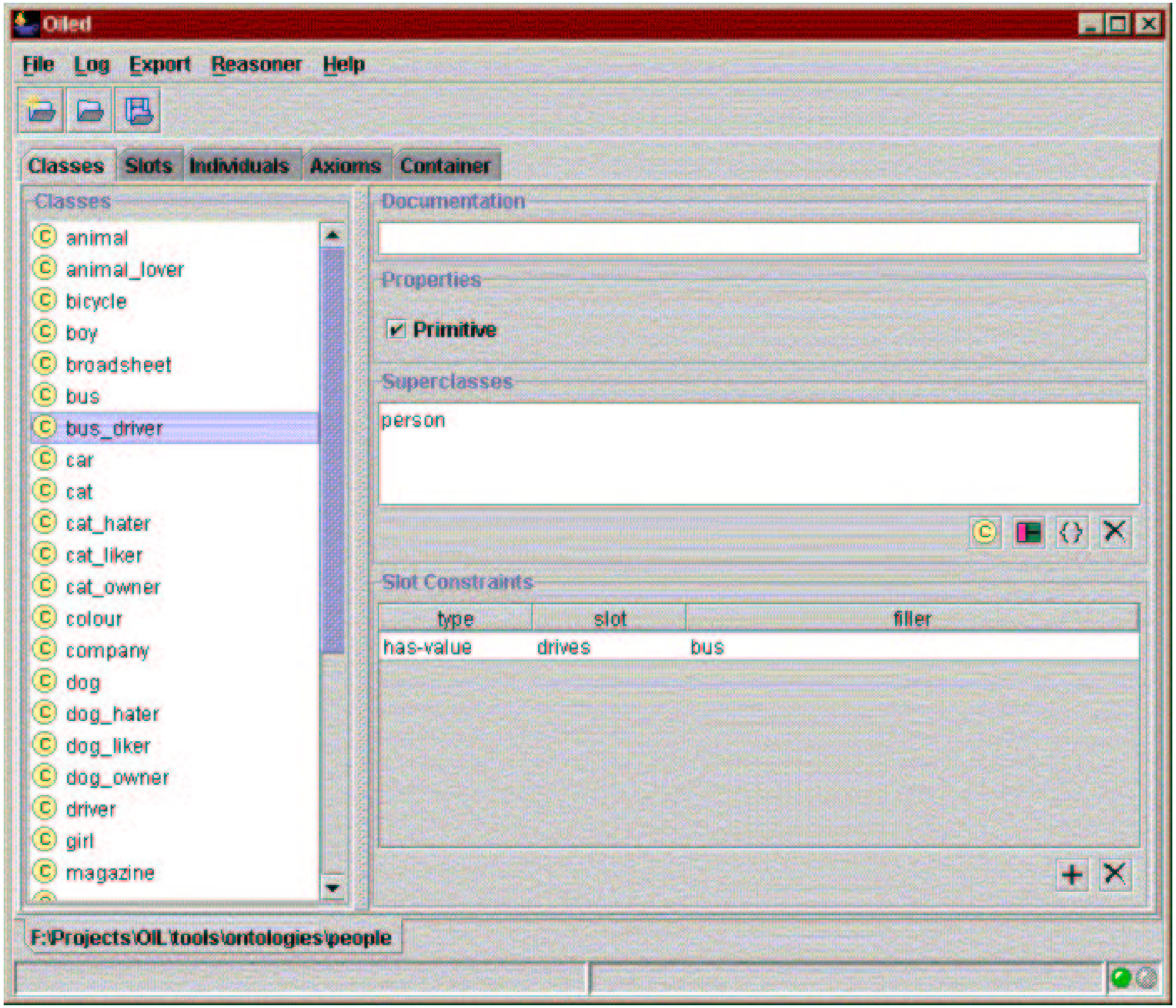
- ☞ **Frame based** interface (inspired by Protégé)
  - Classes defined by superclass(es) plus slot constraints
- ☞ Extended to **clarify semantics** and capture whole language
  - Primitive ($\sqsubseteq$) and defined ($\doteq$) classes
  - Explicit $\exists$ (hasClass), $\forall$ (toClass) and cardinality restrictions
  - Boolean connectives ($\sqcap$, $\sqcup$, $\neg$) and nesting
  - Transitive, symmetrical and functional properties
  - Disjointness, inclusion ($\sqsubseteq$) and equality ($\doteq$) axioms
  - Fake individuals

Reasoning Support for Ontology Design: OilEd

OilEd is a DAML+OIL **ontology editor** with DL reasoning support

- ☞ **Frame based** interface (inspired by Protégé)
  - Classes defined by superclass(es) plus slot constraints
- ☞ Extended to **clarify semantics** and capture whole language
  - Primitive ($\sqsubseteq$) and defined ($\doteq$) classes
  - Explicit $\exists$ (hasClass), $\forall$ (toClass) and cardinality restrictions
  - Boolean connectives ($\sqcap$, $\sqcup$, $\neg$) and nesting
  - Transitive, symmetrical and functional properties
  - Disjointness, inclusion ($\sqsubseteq$) and equality ($\doteq$) axioms
  - Fake individuals
- ☞ **Reasoning support** provided by FaCT system
  - Ontology translated into $\mathcal{SHIQ}$ DL
  - Communicates with FaCT via CORBA interface
  - Indicates inconsistencies and implicit subsumptions

OilEd



OilEd
File
Log
Export
Reasoner
Help
Classes
Slots
Individuals
Axioms
Container
Classes
animal
animal_lover
bicycle
boy
broadsheet
bus
bus_driver
car
cat
cat_hater
cat_liker
cat_owner
colour
company
dog
dog_hater
dog_liker
dog_owner
driver
girl
magazine
Documentation
Properties
Primitive
Superclasses
person
Slot Constraints
type
slot
filler
has-value
drives
bus
F:\Projects\OIL\tools\ontologies\people

Description Logics and Reasoning

What are Description Logics?

What are Description Logics?

A family of logic based Knowledge Representation formalisms

What are Description Logics?

A family of logic based Knowledge Representation formalisms

- ☞ Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects

What are Description Logics?

A family of logic based Knowledge Representation formalisms

- ☞ Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects
- ☞ Descendants of **semantic networks** and **KL-ONE**

What are Description Logics?

A family of logic based Knowledge Representation formalisms

- ☞ Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects
- ☞ Descendants of **semantic networks** and **KL-ONE**
- ☞ **Decidable fragments** of FOL
  - Many DLs are fragments of L2, C2 or the **Guarded Fragment**

What are Description Logics?

A family of logic based Knowledge Representation formalisms

- Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects
- Descendants of **semantic networks** and **KL-ONE**
- **Decidable fragments** of FOL
  - Many DLs are fragments of L2, C2 or the **Guarded Fragment**
- Closely related to **propositional modal logics**

What are Description Logics?

A family of logic based Knowledge Representation formalisms

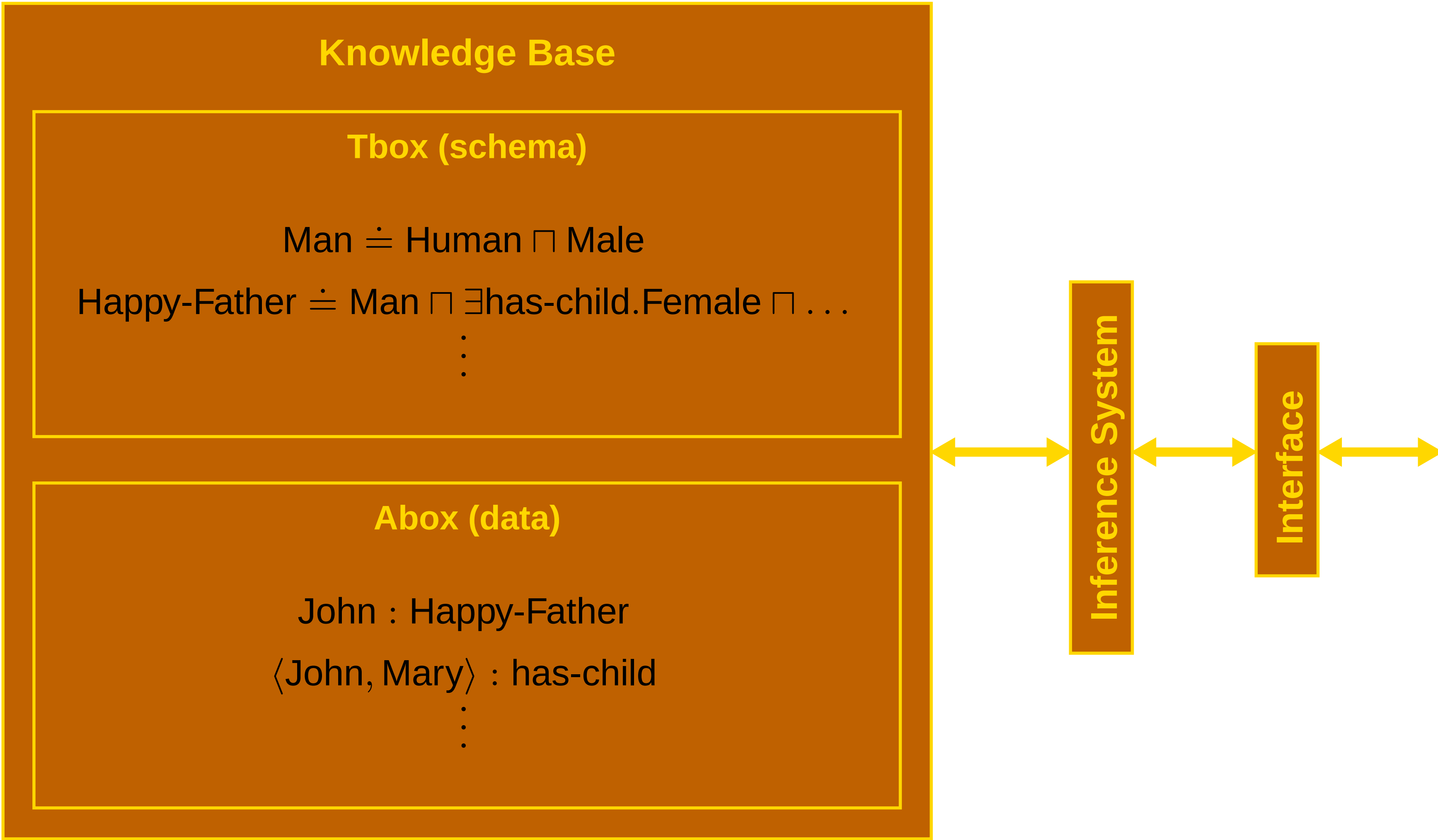
- Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects
- Descendants of **semantic networks** and **KL-ONE**
- **Decidable fragments** of FOL
  - Many DLs are fragments of L2, C2 or the **Guarded Fragment**
- Closely related to **propositional modal logics**
- Also known as terminological logics, concept languages, etc.

What are Description Logics?

A family of logic based Knowledge Representation formalisms

- Based on **concepts** (classes) and **roles**
  - Concepts (classes) are interpreted as sets of objects
  - Roles are interpreted as binary relations on objects
- Descendants of **semantic networks** and **KL-ONE**
- **Decidable fragments** of FOL
  - Many DLs are fragments of L2, C2 or the **Guarded Fragment**
- Closely related to **propositional modal logics**
- Also known as terminological logics, concept languages, etc.
- Key features of DLs are
  - Well defined **semantics** (they are logics)
  - Provision of **inference services**

DL System Architecture



DL Constructors

Particular DLs characterised by **set of constructors** provided for building complex concepts and roles from simpler ones

DL Constructors

Particular DLs characterised by **set of constructors** provided for building complex concepts and roles from simpler ones

☞ Usually include at least:

- Conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)
- Restricted (guarded) forms of quantification ($\exists$, $\forall$)

DL Constructors

Particular DLs characterised by **set of constructors** provided for building complex concepts and roles from simpler ones

- Usually include at least:
  - Conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)
  - Restricted (guarded) forms of quantification ($\exists$, $\forall$)
- This basic DL is known as $\mathcal{ALC}$

DL Constructors

Particular DLs characterised by **set of constructors** provided for building complex concepts and roles from simpler ones

- ☞ Usually include at least:
  - Conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)
  - Restricted (guarded) forms of quantification ($\exists$, $\forall$)
- ☞ This basic DL is known as $\mathcal{ALC}$

For example, concept **Happy Father** in $\mathcal{ALC}$:

$$\begin{array}{lll} \text{Man} & \sqcap & \exists\text{has-child.Male} \\ & \sqcap & \exists\text{has-child.Female} \\ & \sqcap & \forall\text{has-child.}(\text{Doctor} \sqcup \text{Lawyer}) \end{array}$$

DL Syntax and Semantics

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

DL Syntax and Semantics

Semantics given by **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	has-child	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

and for C, D concepts and R a role name

conjunction	$C \sqcap D$	Human $\sqcap$ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	Doctor $\sqcup$ Lawyer	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\neg$Male	$\Delta^{\mathcal{I}} \setminus C$
exists restr.	$\exists R.C$	$\exists$has-child.Male	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restr.	$\forall R.C$	$\forall$has-child.Doctor	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \implies y \in C^{\mathcal{I}}\}$

Other DL Constructors

Many different DLs/DL constructors have been investigated, e.g.

Other DL Constructors

Many different DLs/DL constructors have been investigated, e.g.

Constructor	Syntax	Example	Semantics
qualified num	$\geqslant nR.C$	$\geqslant 3$ child. female	$\{x \mid \lvert\{y.(\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}\rvert \geqslant n\}$
restrictions	$\leqslant nR.C$	$\leqslant 1$ parent female	$\{x \mid \lvert\{y.(\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}\rvert \leqslant n\}$
inverse role	R^{-}	has-child^{-}	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$
trans role	$^{(+)}R$	$^{(+)}$has-ancestor	$R^{\mathcal{I}} = (R^{\mathcal{I}})^{+}$

$$\mathcal{SHIQ}$$

nominals	$\{x\}$	{Italy}	$\{x^{\mathcal{I}}\}$
conc. domain	$f_1,\ldots,f_n.P$	earns spends $<$	$\{x \mid P(f_1^{\mathcal{I}},\ldots,f_n^{\mathcal{I}})\}$

$$\mathcal{SHOIQ}(D_n)$$

$\vdots$

DL Knowledge Base (Tbox)

Terminological part (**Tbox**) is set of axioms describing **structure** of domain

DL Knowledge Base (Tbox)

Terminological part (**Tbox**) is set of axioms describing **structure** of domain

Definition axioms introduce macros/names for concepts

$A \doteq C$, $A \sqsubseteq C$

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

Human $\sqsubseteq$ Animal $\sqcap$ Biped

DL Knowledge Base (Tbox)

Terminological part (**Tbox**) is set of axioms describing **structure** of domain

Definition axioms introduce macros/names for concepts

$A \doteq C$, $A \sqsubseteq C$

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

Human $\sqsubseteq$ Animal $\sqcap$ Biped

Inclusion (GCI) axioms assert subsumption relations

$C \sqsubseteq D$ (note $C \doteq D$ equivalent to $C \sqsubseteq D$ and $D \sqsubseteq C$)

$\exists$has-degree.Masters $\sqsubseteq$ $\exists$has-degree.Bachelors

DL Knowledge Base (Tbox)

Terminological part (**Tbox**) is set of axioms describing **structure** of domain

Definition axioms introduce macros/names for concepts

$A \doteq C$, $A \sqsubseteq C$

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

Human $\sqsubseteq$ Animal $\sqcap$ Biped

Inclusion (GCI) axioms assert subsumption relations

$C \sqsubseteq D$ (note $C \doteq D$ equivalent to $C \sqsubseteq D$ and $D \sqsubseteq C$)

$\exists$has-degree.Masters $\sqsubseteq$ $\exists$has-degree.Bachelors

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

$C \doteq D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

A **Tbox** $\mathcal{T}$ iff it satisfies every axiom in $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$)

DL Knowledge Base (Abox)

Assertional part (**Abox**) is set of axioms describing **concrete situation**

DL Knowledge Base (Abox)

Assertional part (**Abox**) is set of axioms describing **concrete situation**

Concept assertions

$a : C$

John : Man $\sqcap$ $\exists$has-child.Female

DL Knowledge Base (Abox)

Assertional part (**Abox**) is set of axioms describing **concrete situation**

Concept assertions

$a : C$

John : Man $\sqcap$ $\exists$has-child.Female

Role assertions

$\langle a, b \rangle : R$

$\langle$John, Mary$\rangle$: has-child

DL Knowledge Base (Abox)

Assertional part (**Abox**) is set of axioms describing **concrete situation**

Concept assertions

$a : C$

John : Man $\sqcap$ $\exists$has-child.Female

Role assertions

$\langle a, b \rangle : R$

$\langle$John, Mary$\rangle$: has-child

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

$a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ $\qquad$ $\langle a, b \rangle : R$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$

An **Abox** $\mathcal{A}$ iff it satisfies every axiom in $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$)

A **KB** $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ iff it satisfies both $\mathcal{T}$ and $\mathcal{A}$ ($\mathcal{I} \models \Sigma$)

Why Tbox and Abox?

- Restricted use of individuals maintains (kind of) **tree model property**
  - Arbitrary but finite directed graph connecting named individuals
  - Named individuals roots of (possibly) infinite trees of anonymous individuals
  - Lower complexity class (ExpTime for $\mathcal{SHIQ}$)
  - Easier to design and optimise (tableaux) algorithms

Why Tbox and Abox?

- ☞ Restricted use of individuals maintains (kind of) **tree model property**
  - Arbitrary but finite directed graph connecting named individuals
  - Named individuals roots of (possibly) infinite trees of anonymous individuals
  - Lower complexity class (ExpTime for $\mathcal{SHIQ}$)
  - Easier to design and optimise (tableaux) algorithms
- ☞ Existentially defined classes (nominals) destroy this property
  - Trees can “loop back” to named individuals
  - Higher complexity class (NExpTime for $\mathcal{SHIQ}$)
  - No known tableaux algorithm for $\mathcal{SHIQ}$ + nominals

Why Tbox and Abox?

- Restricted use of individuals maintains (kind of) **tree model property**
  - Arbitrary but finite directed graph connecting named individuals
  - Named individuals roots of (possibly) infinite trees of anonymous individuals
  - Lower complexity class (ExpTime for $\mathcal{SHIQ}$)
  - Easier to design and optimise (tableaux) algorithms
- Existentially defined classes (nominals) destroy this property
  - Trees can "loop back" to named individuals
  - Higher complexity class (NExpTime for $\mathcal{SHIQ}$)
  - No known tableaux algorithm for $\mathcal{SHIQ}$ + nominals
- Note that with nominals, Abox becomes syntactic sugar
  - $a : C$ equiv. to $\{a\} \sqsubseteq C$
  - $\langle a, b \rangle : R$ equiv. to $\{a\} \sqsubseteq \exists R.\{b\}$

Basic Inference Problems

Basic Inference Problems

Subsumption (structure knowledge, compute taxonomy)

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

Basic Inference Problems

Subsumption (structure knowledge, compute taxonomy)

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

Subsumption w.r.t. Tbox $\mathcal{T}$

$C \sqsubseteq_{\mathcal{T}} D$? Is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models of $\mathcal{T}$?

Basic Inference Problems

Subsumption (structure knowledge, compute taxonomy)

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

Subsumption w.r.t. Tbox $\mathcal{T}$

$C \sqsubseteq_{\mathcal{T}} D$? Is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models of $\mathcal{T}$?

Consistency

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

Basic Inference Problems

Subsumption (structure knowledge, compute taxonomy)

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

Subsumption w.r.t. Tbox $\mathcal{T}$

$C \sqsubseteq_{\mathcal{T}} D$? Is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models of $\mathcal{T}$?

Consistency

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

KB Consistency

Is $\langle \mathcal{T}, \mathcal{A} \rangle$ consistent? Is there a model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{A} \rangle$?

Basic Inference Problems

Subsumption (structure knowledge, compute taxonomy)

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

Subsumption w.r.t. Tbox $\mathcal{T}$

$C \sqsubseteq_\mathcal{T} D$? Is $C^\mathcal{I} \subseteq D^\mathcal{I}$ in all models of $\mathcal{T}$?

Consistency

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

KB Consistency

Is $\langle \mathcal{T}, \mathcal{A} \rangle$ consistent? Is there a model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{A} \rangle$?

Problems are **closely related**:

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

Reasoning Techniques

Subsumption and Satisfiability

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- ☞ Try to build **model** (witness) of concept C

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- ☞ Try to build **model** (witness) of concept C
- ☞ Model represented by **tree** $\mathbf{T}$
  - Nodes in $\mathbf{T}$ correspond to individuals in model
  - Nodes labeled with sets of subconcepts of C
  - Edges labeled with role names in C

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- ☞ Try to build **model** (witness) of concept C
- ☞ Model represented by **tree** $\mathbf{T}$
  - Nodes in $\mathbf{T}$ correspond to individuals in model
  - Nodes labeled with sets of subconcepts of C
  - Edges labeled with role names in C
- ☞ Start from **root node** labeled $\{C\}$

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- ☞ Try to build **model** (witness) of concept C
- ☞ Model represented by **tree** $\mathbf{T}$
  - Nodes in $\mathbf{T}$ correspond to individuals in model
  - Nodes labeled with sets of subconcepts of C
  - Edges labeled with role names in C
- ☞ Start from **root node** labeled $\{C\}$
- ☞ Apply **expansion rules** to node labels until
  - Rules correspond with language constructs
  - Expansion completed (tree represents valid model)
  - Contradictions prove there is no model

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- ☞ Try to build **model** (witness) of concept C
- ☞ Model represented by **tree** $\mathbf{T}$
  - Nodes in $\mathbf{T}$ correspond to individuals in model
  - Nodes labeled with sets of subconcepts of C
  - Edges labeled with role names in C
- ☞ Start from **root node** labeled $\{C\}$
- ☞ Apply **expansion rules** to node labels until
  - Rules correspond with language constructs
  - Expansion completed (tree represents valid model)
  - Contradictions prove there is no model
- ☞ Non-deterministic expansion $\longrightarrow$ **search** (e.g., $C \sqcup D$)

Subsumption and Satisfiability

Subsumption transformed into **satisfiability**

Tableaux algorithm used to test satisfiability

- Try to build **model** (witness) of concept C
- Model represented by **tree** $\mathbf{T}$
  - Nodes in $\mathbf{T}$ correspond to individuals in model
  - Nodes labeled with sets of subconcepts of C
  - Edges labeled with role names in C
- Start from **root node** labeled $\{C\}$
- Apply **expansion rules** to node labels until
  - Rules correspond with language constructs
  - Expansion completed (tree represents valid model)
  - Contradictions prove there is no model
- Non-deterministic expansion $\longrightarrow$ **search** (e.g., $C \sqcup D$)
- **Blocking** ensures termination (with expressive DLs)

Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

Tableaux Expansion

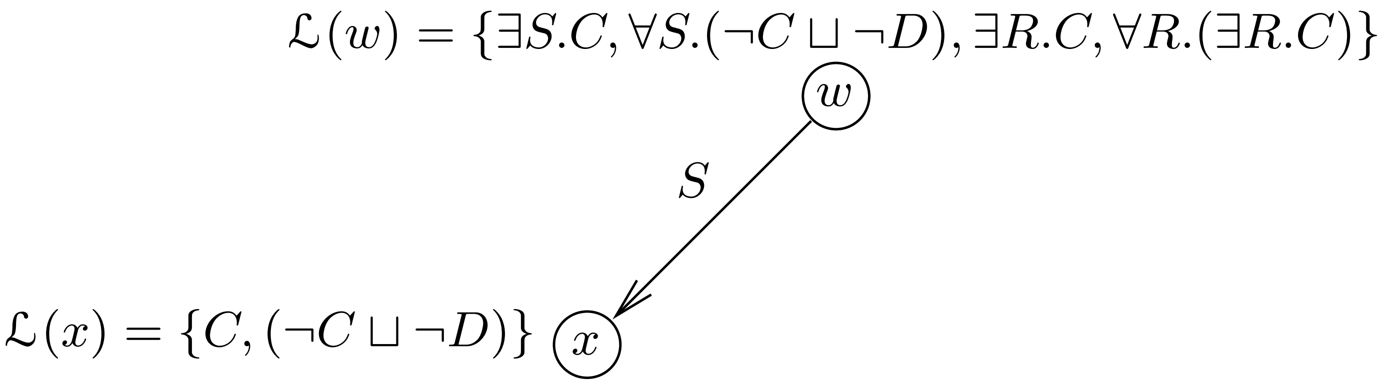
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

$$w$$

Tableaux Expansion

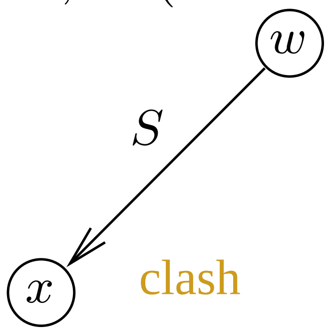
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role

$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$

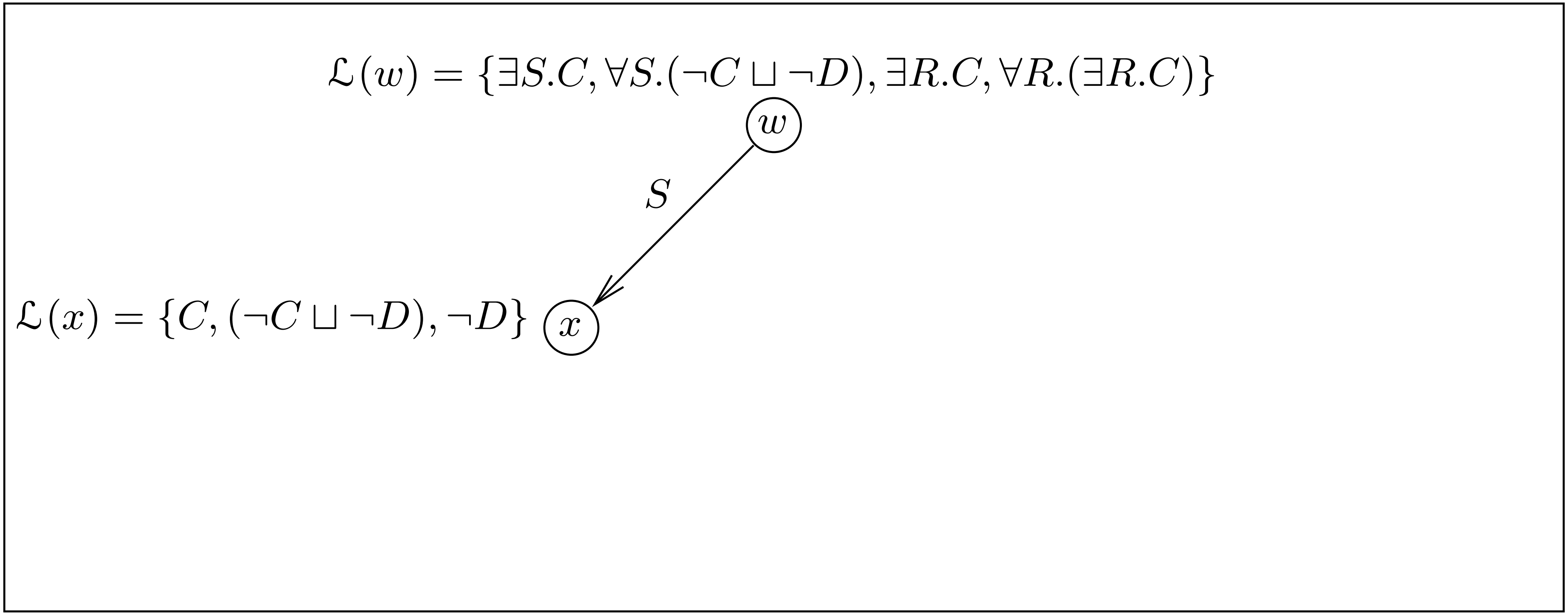


$$\mathcal{L}(x) = \{C, (\neg C \sqcup \neg D), \neg C\}$$

clash

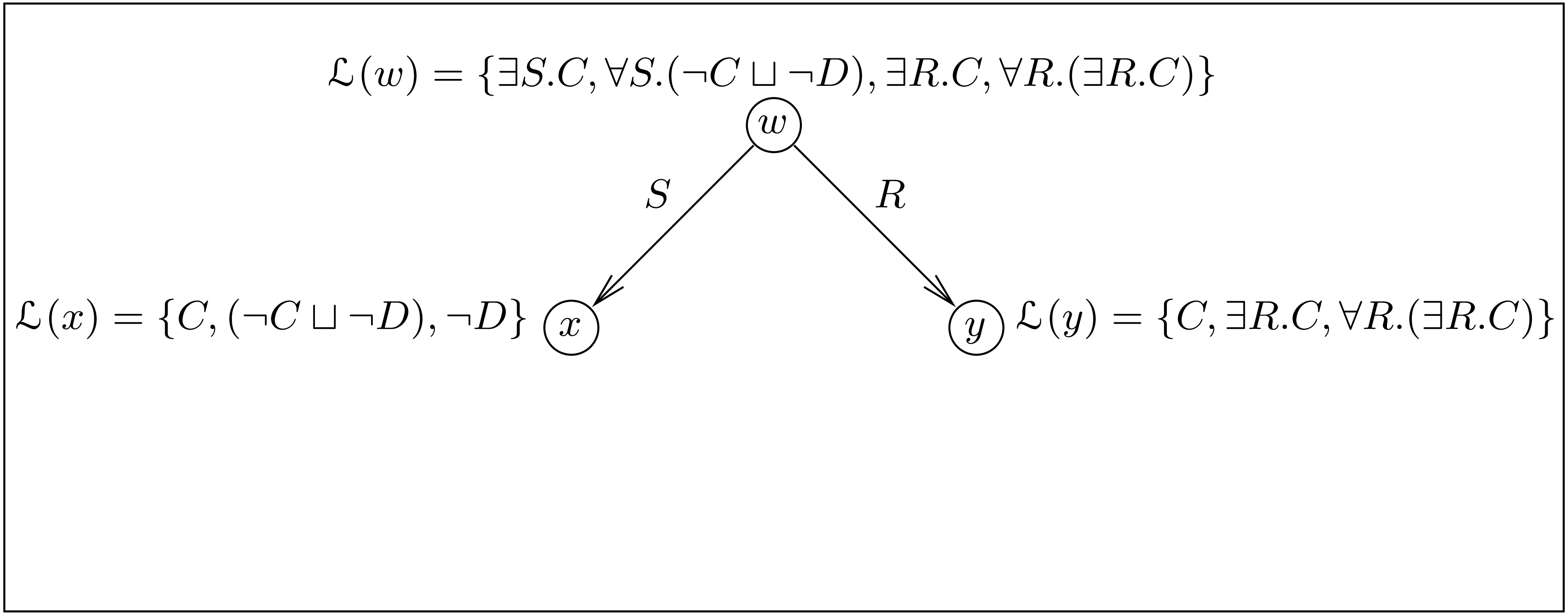
Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



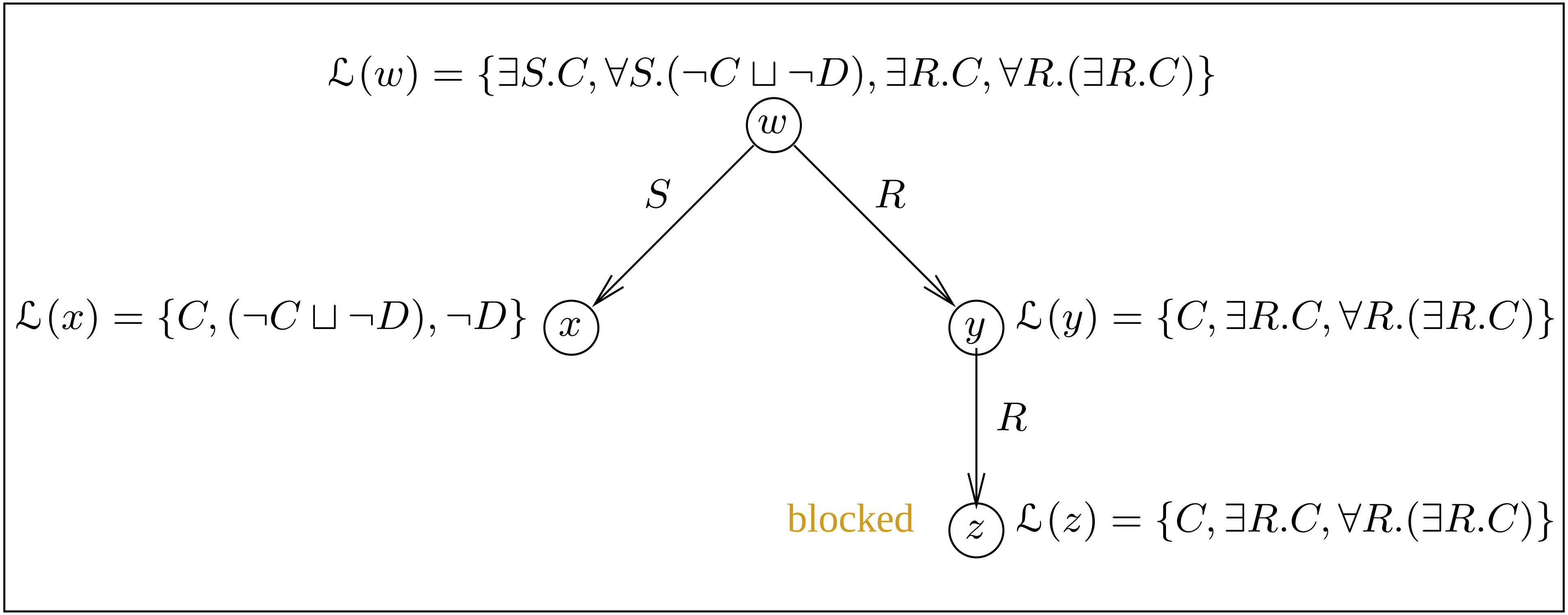
Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



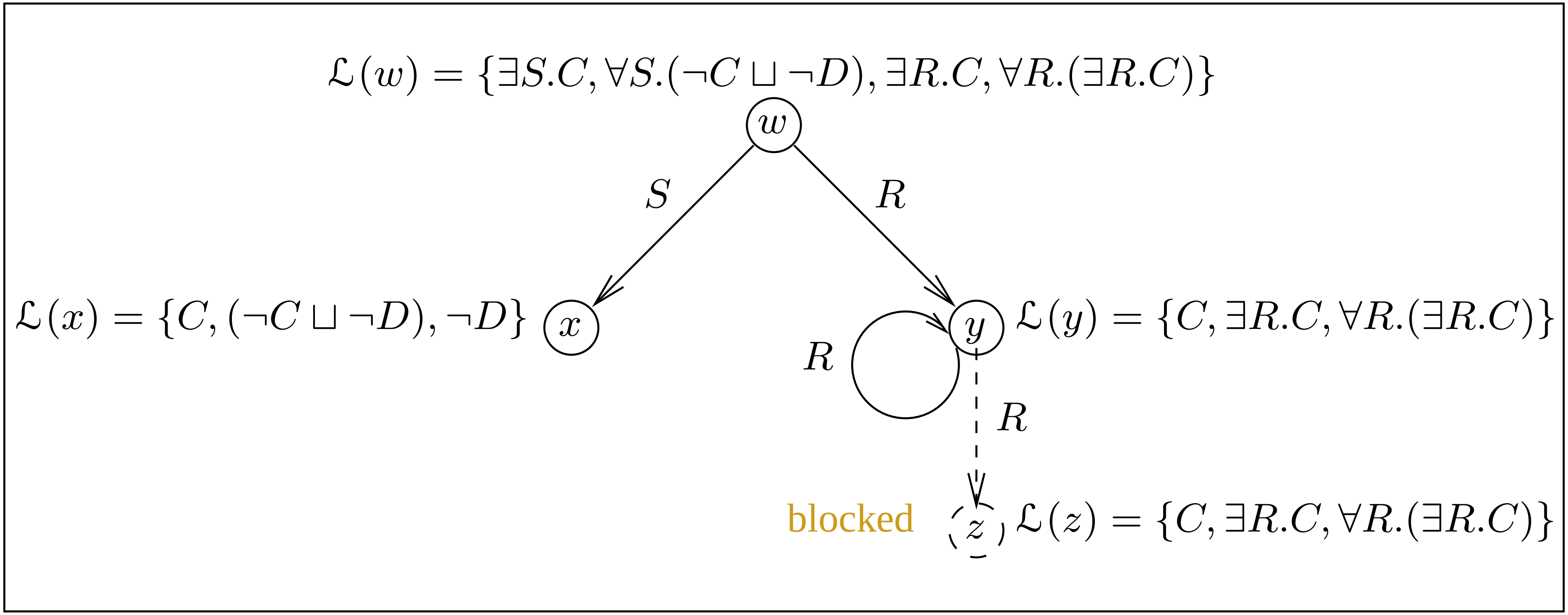
Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Expansion

Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Concept is **satisfiable**: w is a **witness**

More Advanced Techniques

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- For each GCI $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to **every** node label

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- For each GCI $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to **every** node label

More expressive DLs

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- For each GCI $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to **every** node label

More expressive DLs

- Basic technique can be extended to deal with
  - Role inclusion axioms (role hierarchy)
  - Number restrictions
  - Inverse roles
  - Concrete domains
  - Aboxes
  - etc.

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- ☞ For each GCI $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to **every** node label

More expressive DLs

- ☞ Basic technique can be extended to deal with
  - Role inclusion axioms (role hierarchy)
  - Number restrictions
  - Inverse roles
  - Concrete domains
  - Aboxes
  - etc.
- ☞ Extend **expansion rules** and use more sophisticated **blocking** strategy

More Advanced Techniques

Satisfiability w.r.t. a Terminology

☞ For each GCI $C \sqsubseteq D \in \mathcal{T}$, add $\neg C \sqcup D$ to **every** node label

More expressive DLs

☞ Basic technique can be extended to deal with
- Role inclusion axioms (role hierarchy)
- Number restrictions
- Inverse roles
- Concrete domains
- Aboxes
- etc.

☞ Extend **expansion rules** and use more sophisticated **blocking** strategy

☞ Forest instead of Tree (for Aboxes)

Implementing DL Systems

Naive Implementations

Problems include:

Naive Implementations

Problems include:

- ☞ **Space** usage

Naive Implementations

Problems include:

- ☞ **Space** usage
  - Storage required for tableaux datastructures

Naive Implementations

Problems include:

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

Naive Implementations

Problems include:

- ☞ **Space** usage
  - Storage required for tableaux datastructures
  - Rarely a serious problem in practice
  - But problems can arise with inverse roles and cyclical KBs

Naive Implementations

Problems include:

- ☞ **Space** usage
  - Storage required for tableaux datastructures
  - Rarely a serious problem in practice
  - But problems can arise with inverse roles and cyclical KBs
- ☞ **Time** usage

Naive Implementations

Problems include:

- **Space** usage
  - Storage required for tableaux datastructures
  - Rarely a serious problem in practice
  - But problems can arise with inverse roles and cyclical KBs
- **Time** usage
  - Search required due to non-deterministic expansion

Naive Implementations

Problems include:

☞ **Space** usage

- Storage required for tableaux datastructures
- Rarely a serious problem in practice
- But problems can arise with inverse roles and cyclical KBs

☞ **Time** usage

- Search required due to non-deterministic expansion
- **Serious** problem in practice

Naive Implementations

Problems include:

- ☞ **Space** usage
  - Storage required for tableaux datastructures
  - Rarely a serious problem in practice
  - But problems can arise with inverse roles and cyclical KBs
- ☞ **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

Careful Choice of Algorithm

- **Transitive roles** instead of transitive closure

Careful Choice of Algorithm

☞ **Transitive roles** instead of transitive closure

- Deterministic expansion of $\exists R.C$, even when $R \in \mathbf{R}_+$

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

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) valid cyclical models

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) valid cyclical models
- **Direct algorithm**/implementation instead of encodings

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) valid cyclical models

☞ **Direct algorithm**/implementation instead of encodings
- GCI axioms can be used to “encode” additional operators/axioms

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) valid 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

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) valid 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, . . .

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) valid 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., $(\text{domain } R.C) \equiv \exists R.\top \sqsubseteq C$

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) valid 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

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) valid 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

Modern systems include **MANY** optimisations, e.g.:

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing
  - Normalisation and simplification of concepts

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- Optimised **subsumption** testing
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms
  - Davis-Putnam style semantic branching search

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms
  - Davis-Putnam style semantic branching search
  - Dependency directed backtracking

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms
  - Davis-Putnam style semantic branching search
  - Dependency directed backtracking
  - Caching

Highly Optimised Implementation

Modern systems include **MANY** optimisations, e.g.:

- ☞ Optimised **classification**
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms
  - Davis-Putnam style semantic branching search
  - Dependency directed backtracking
  - Caching
  - Heuristic ordering of propositional and modal expansion

Dependency Directed Backtracking

Dependency Directed Backtracking

☞ Allows **rapid recovery** from bad branching choices

Dependency Directed Backtracking

- Allows **rapid recovery** from bad branching choices
- Most commonly used technique is **backjumping**

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)

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**

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

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

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

Dependency Directed Backtracking

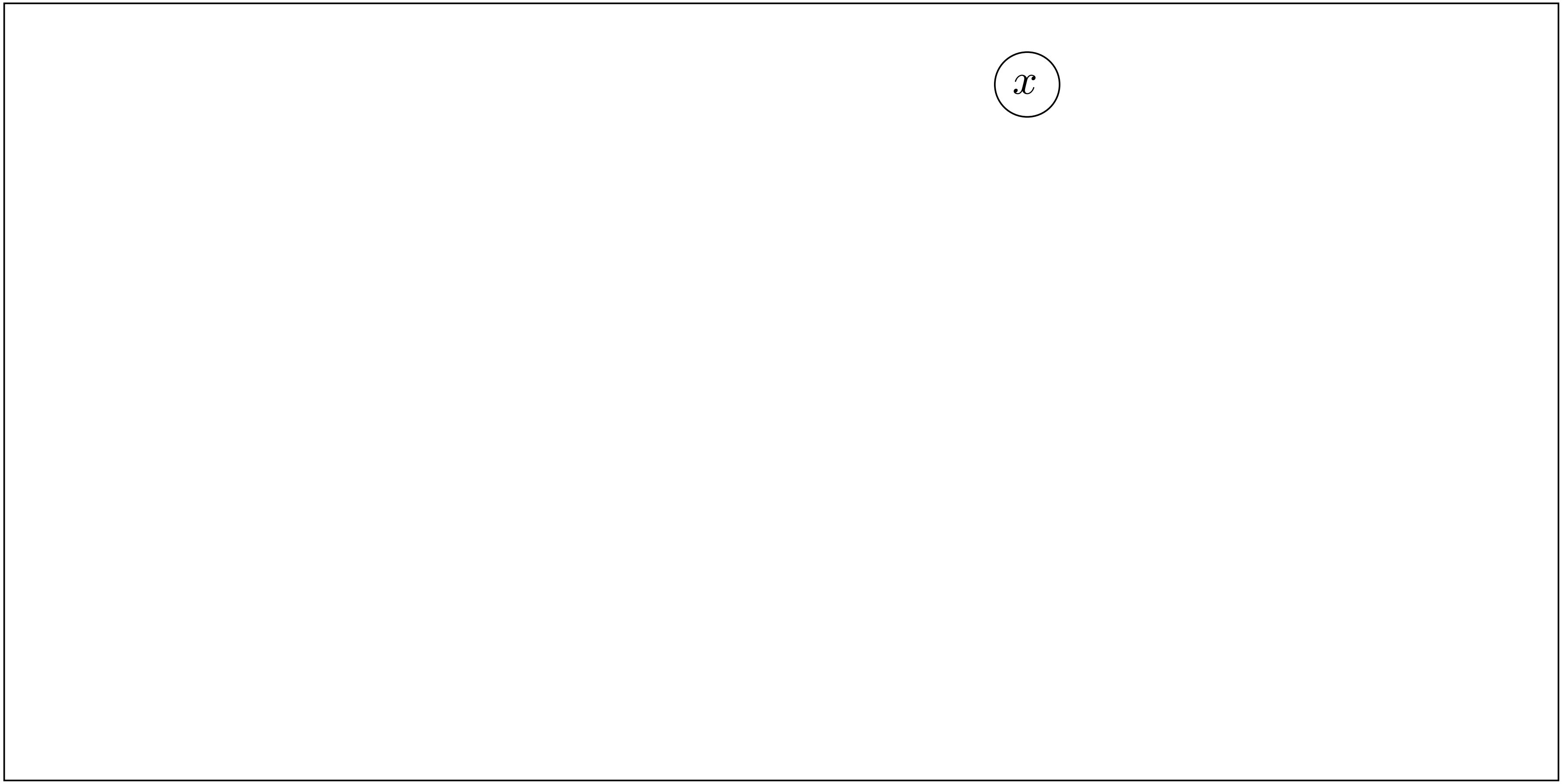
- Allows **rapid recovery** from bad branching choices
- Most commonly used technique is **backjumping**
  - Tag concepts introduced at **branch points** (e.g., when expanding disjunctions)
  - Expansion rules combine and **propagate tags**
  - On discovering a clash, **identify** most recently introduced concepts involved
  - **Jump back** to relevant branch points **without exploring** alternative branches
  - Effect is to **prune** away part of the search space
- **Highly effective** — essential for usable system
  - E.g., GALEN KB, 30s (with) $\longrightarrow$ months++ (without)

Backjumping

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

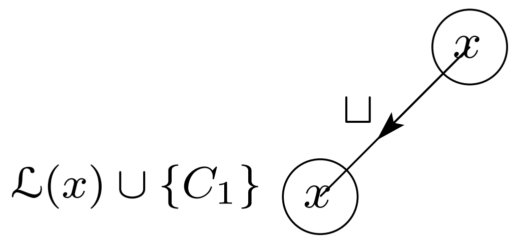
Backjumping

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



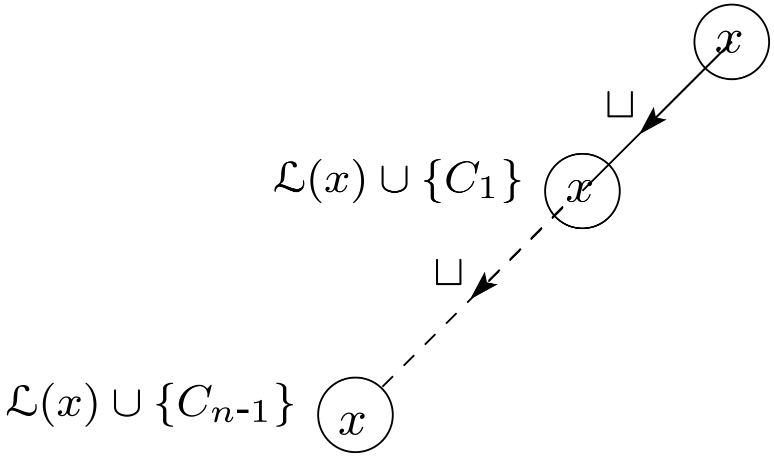
Backjumping

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



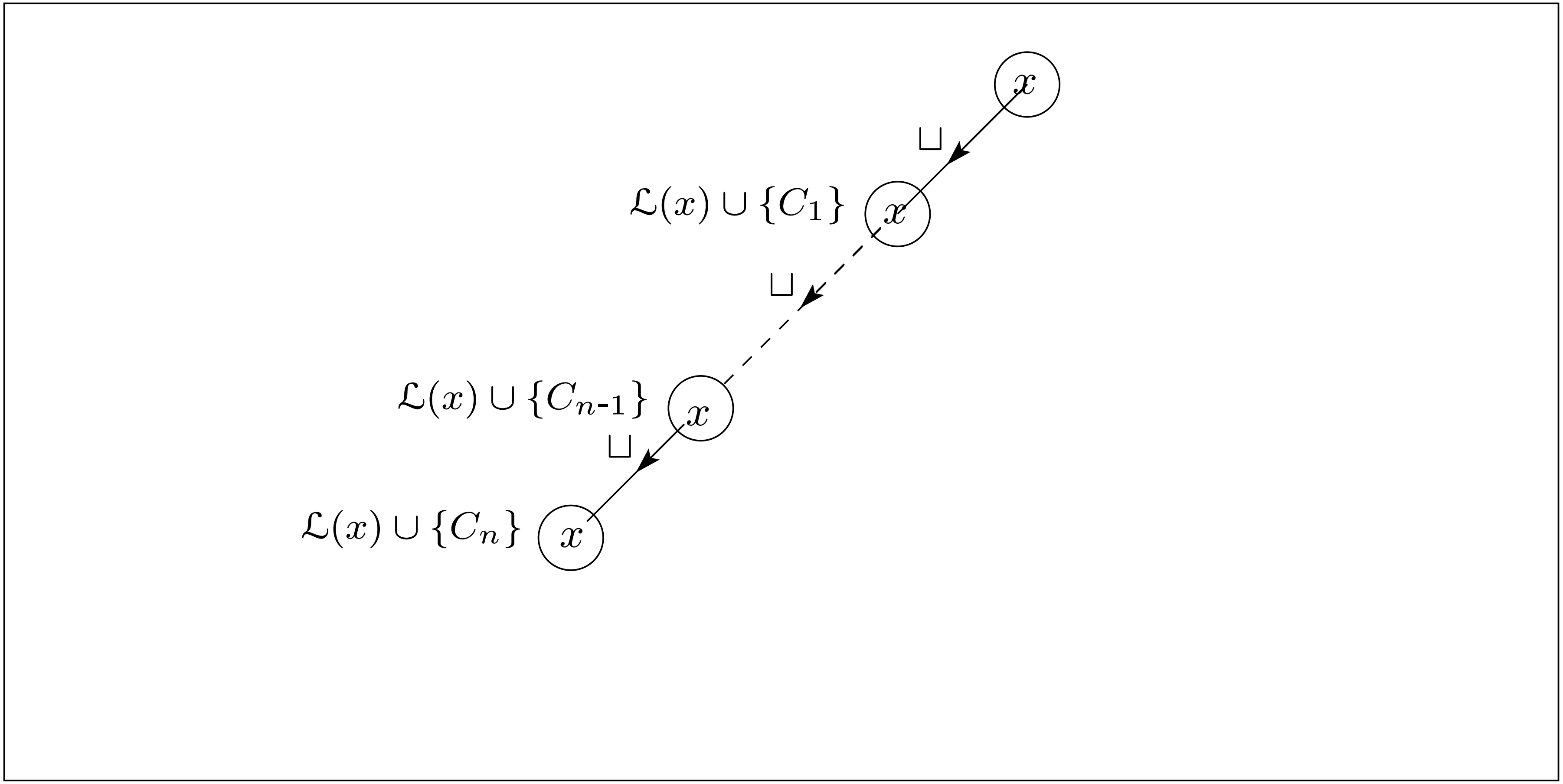
Backjumping

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



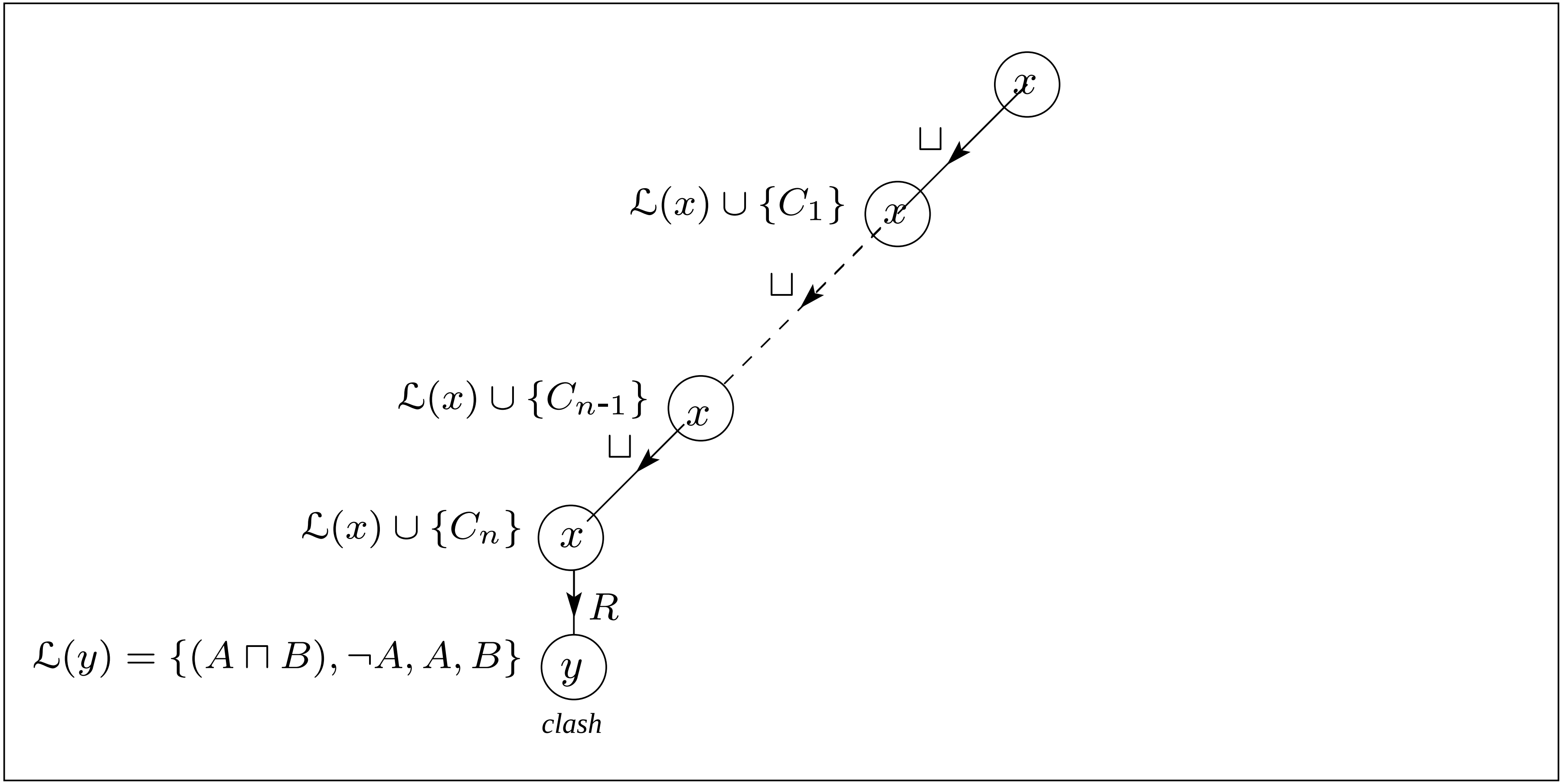
Backjumping

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



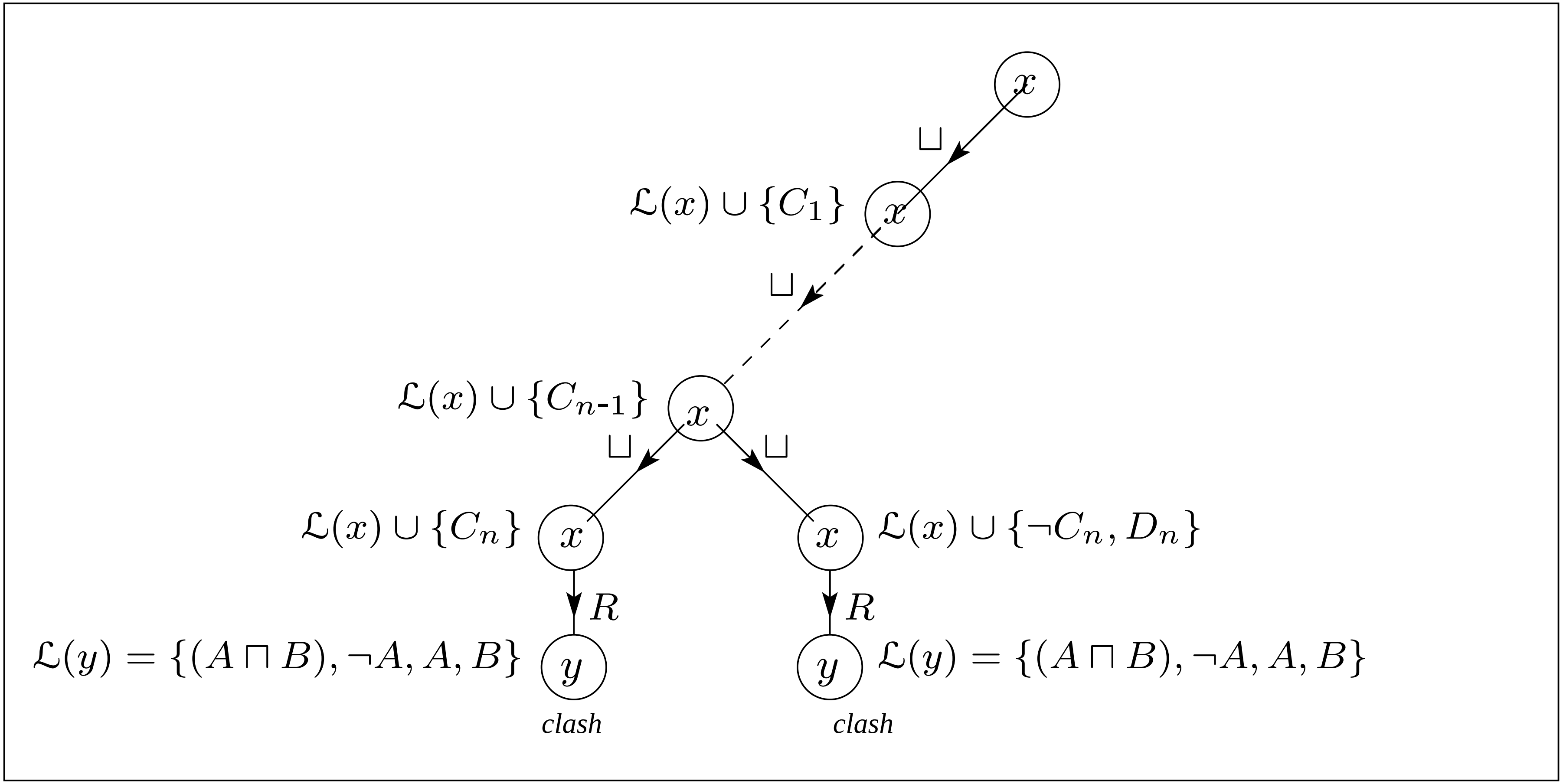
Backjumping

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



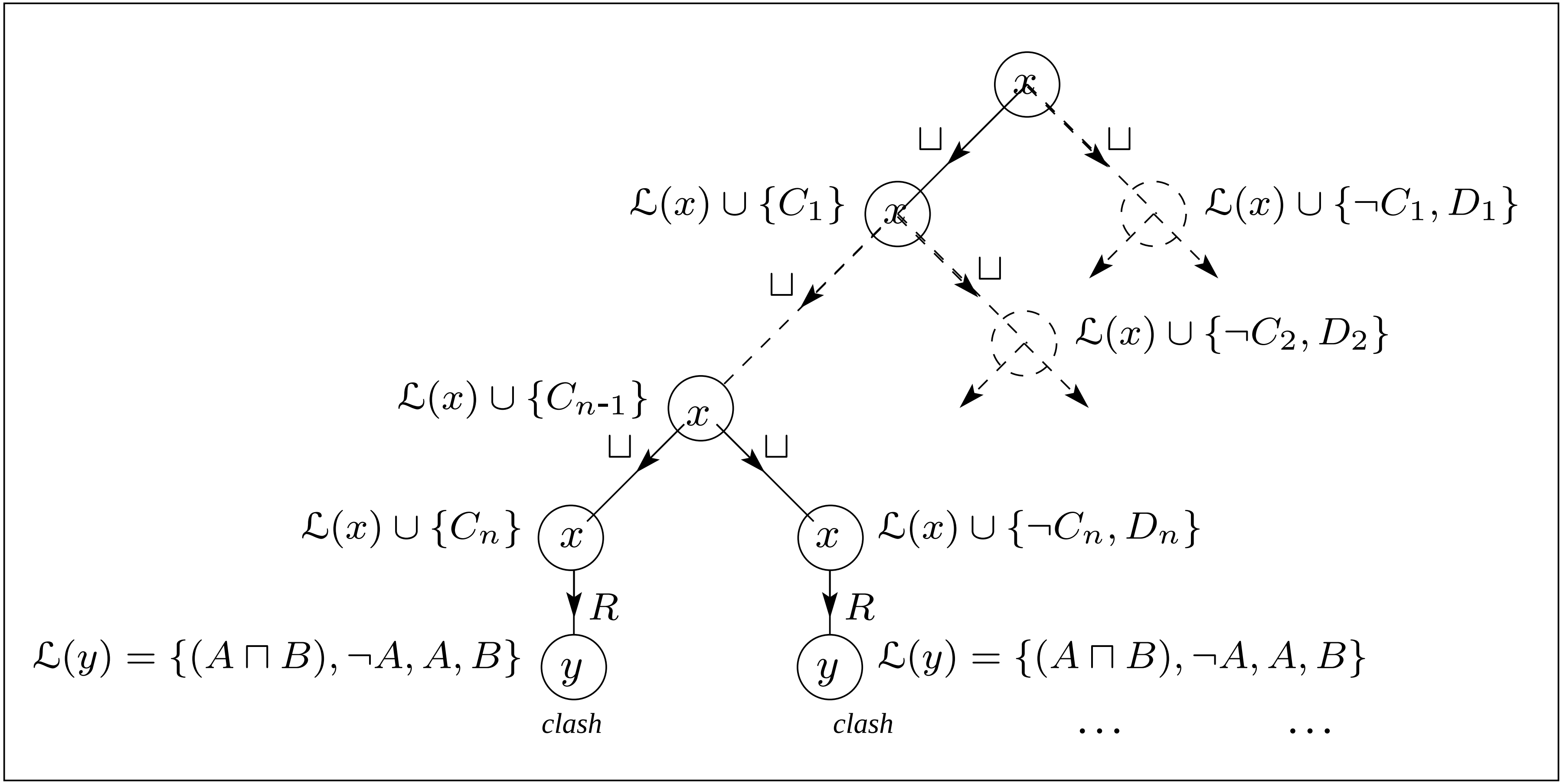
Backjumping

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



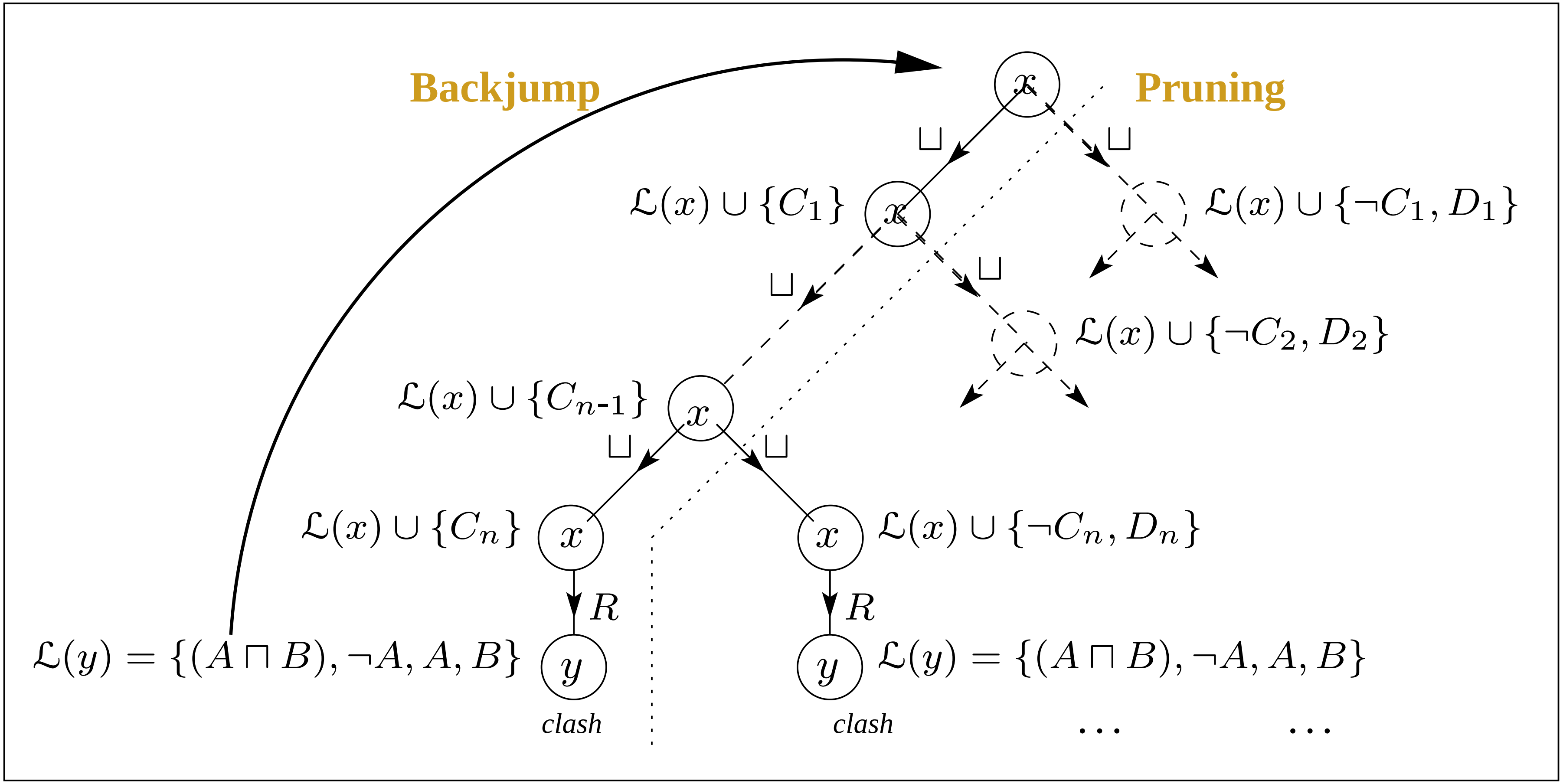
Backjumping

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



Backjumping

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



Axioms and Rules

KR Rules (Horn Clauses)

☞ Rules (at least KR rules) can be seen as a form of axiom, e.g.:

$$\begin{aligned} p(x) \leftarrow q(x) \wedge w(x) \quad &\equiv \quad p \sqsubseteq q \sqcap w \\ p(x) \leftarrow q(x) \wedge r(x,y) \wedge w(y) \quad &\equiv \quad p \sqsubseteq q \sqcap \exists r.w \end{aligned}$$

KR Rules (Horn Clauses)

- Rules (at least KR rules) can be seen as a form of axiom, e.g.:

$$p(x) \leftarrow q(x) \wedge w(x) \quad \equiv \quad p \sqsubseteq q \sqcap w$$
$$p(x) \leftarrow q(x) \wedge r(x,y) \wedge w(y) \quad \equiv \quad p \sqsubseteq q \sqcap \exists r.w$$

- Distinguished variables have implicit $\forall$, others have implicit $\exists$, i.e.:

$$p(x) \leftarrow q(x) \wedge r(x,y) \quad \equiv \quad \forall x(p(x) \leftarrow (\exists y(q(x) \wedge r(x,y))))$$

KR Rules (Horn Clauses)

- Rules (at least KR rules) can be seen as a form of axiom, e.g.:

$$\begin{aligned} p(x) \leftarrow q(x) \wedge w(x) \quad &\equiv \quad p \sqsubseteq q \sqcap w \\ p(x) \leftarrow q(x) \wedge r(x,y) \wedge w(y) \quad &\equiv \quad p \sqsubseteq q \sqcap \exists r.w \end{aligned}$$

- Distinguished variables have implicit $\forall$, others have implicit $\exists$, i.e.:

$$p(x) \leftarrow q(x) \wedge r(x,y) \quad \equiv \quad \forall x(p(x) \leftarrow (\exists y(q(x) \wedge r(x,y))))$$

- Closed world doesn't make sense in ontologies
  - **Don't** want to infer Person $\sqsubseteq$ American just because only have information about Americans

More Complex Examples

☞ E.g., the "discount" example:

$$\begin{aligned} \text{discount}(x, 7\%) \quad \leftarrow \quad & \text{customer}(x) \wedge \text{category}(x, y) \\ & \wedge \text{premium}(y) \wedge \text{buys}(x, z) \wedge \text{product}(z) \\ & \wedge \text{category}(z, w) \wedge \text{luxury}(w) \end{aligned}$$

can be written in DL as:

$$\begin{aligned} \exists\text{discount}.7\% \quad \sqsubseteq \quad & \text{customer} \sqcap \exists\text{category.premium} \\ & \sqcap \exists\text{buys}.(\text{product} \sqcap \exists\text{category.luxury}) \end{aligned}$$

More Complex Examples

☞ E.g., the "discount" example:

$$\begin{aligned}\text{discount}(x, 7\%) \quad \leftarrow \quad & \text{customer}(x) \wedge \text{category}(x, y) \\ & \wedge \text{premium}(y) \wedge \text{buys}(x, z) \wedge \text{product}(z) \\ & \wedge \text{category}(z, w) \wedge \text{luxury}(w)\end{aligned}$$

can be written in DL as:

$$\begin{aligned}\exists\text{discount}.7\% \quad \sqsubseteq \quad & \text{customer} \sqcap \exists\text{category.premium} \\ & \sqcap \exists\text{buys}.(\text{product} \sqcap \exists\text{category.luxury})\end{aligned}$$

☞ May **not** capture intended semantics

- Should be able to fix this by modeling transactions instead of customers

Query Rules

- Query rules have a completely different semantics

$$(x) \leftarrow q(x) \wedge r(x, y)$$

says answer = $\{x | KB \models \exists y (q(x) \wedge r(x, y))\}$

Query Rules

- Query rules have a completely different semantics

$$(x) \leftarrow q(x) \wedge r(x,y)$$

says answer = $\{x | KB \models \exists y(q(x) \wedge r(x,y))\}$

- Can also reduce this to a standard DL retrieval Query:

retrieve instances of $(p \wedge \exists r.q)$

says answer = $\{x | KB \models \exists y(q(x) \wedge r(x,y))\}$

Query Rules

- Query rules have a completely different semantics

$$(x) \leftarrow q(x) \wedge r(x,y)$$

  says answer = $\{x|KB \models \exists y(q(x) \wedge r(x,y))\}$

- Can also reduce this to a standard DL retrieval Query:

  retrieve instances of $(p \wedge \exists r.q)$

  says answer = $\{x|KB \models \exists y(q(x) \wedge r(x,y))\}$

- Applications can implement many “rule-like” features using queries

What (horn) Rules Can't Capture?

Horn rules with no extensions (probably) can't capture:

- Negation

What (horn) Rules Can't Capture?

Horn rules with no extensions (probably) can't capture:

- Negation
- Disjunction (?)

What (horn) Rules Can't Capture?

Horn rules with no extensions (probably) can't capture:

- ☞ Negation
- ☞ Disjunction (?)
- ☞ $\forall$ in body of rule

What (horn) Rules Can’t Capture?

Horn rules with no extensions (probably) can’t capture:

- Negation
- Disjunction (?)
- $\forall$ in body of rule
- $\exists$ in head of rule

What (horn) Rules Can't Capture?

Horn rules with no extensions (probably) can't capture:

- Negation
- Disjunction (?)
- $\forall$ in body of rule
- $\exists$ in head of rule
- Counting/cardinality constraints

What (horn) Rules Can't Capture?

Horn rules with no extensions (probably) can't capture:

- Negation
- Disjunction (?)
- $\forall$ in body of rule
- $\exists$ in head of rule
- Counting/cardinality constraints

. . . ?

What (standard) DLs Can't Capture

- nary predicates ($n > 2$)
  - but $\mathcal{DLR}$ is an nary DL used in DB applications

What (standard) DLs Can't Capture

- nary predicates ($n > 2$)
  - but $\mathcal{DLR}$ is an nary DL used in DB applications
- Rules that break tree model property, e.g.,

$$\text{uncle}(x, z) \quad \leftarrow \quad \text{parent}(x, y) \wedge \text{brother}(y, z)$$

  - but some (otherwise weak) DLs have function chain equivalence, i.e.,

$$f_1 \circ \ldots \circ f_n \equiv f'_1 \circ \ldots \circ f'_m$$

What (standard) DLs Can't Capture

- nary predicates ($n > 2$)
  - but $\mathcal{DLR}$ is an nary DL used in DB applications
- Rules that break tree model property, e.g.,

$$\text{uncle}(x, z) \quad \leftarrow \quad \text{parent}(x, y) \wedge \text{brother}(y, z)$$

  - but some (otherwise weak) DLs have function chain equivalence, i.e.,

$$f_1 \circ \ldots \circ f_n \equiv f'_1 \circ \ldots \circ f'_m$$

- Can't combine with expressive DLs (and still stay decidable)
  - adding these constructs to $\mathcal{SHIQ}$ leads to undecidability

Intersection of Rules and DLs

☞ Can express horn clauses with:

- conjunction in head ($\equiv$ multiple rules)
- $\forall$ in head
- $\exists$ in body
- only unary or binary predicates
- “inverse” roles/predicates

Intersection of Rules and DLs

☞ Can express horn clauses with:

- conjunction in head ($\equiv$ multiple rules)
- $\forall$ in head
- $\exists$ in body
- only unary or binary predicates
- “inverse” roles/predicates

☞ Result is a strange and asymmetrical DL

Other Approaches

☞ Can layer rules on top of DL
  - rule predicates can be DL classes or roles
  - several examples have been implemented
  - best known is Carin system from Levy & Rousset
  - undecidable unless DL is very weak (Carin uses Classic)

Other Approaches

- Can layer rules on top of DL
  - rule predicates can be DL classes or roles
  - several examples have been implemented
  - best known is Carin system from Levy & Rousset
  - undecidable unless DL is very weak (Carin uses Classic)
- Some existing work on language **fusions** and **hybrid** reasoners

Research Challenges

Research Challenges

- **Increased expressive power**
  - Datatypes
  - Nominals
  - Extensions to DAML+OIL

Research Challenges

- **Increased expressive power**
  - Datatypes
  - Nominals
  - Extensions to DAML+OIL
- **Performance**
  - Inverse roles and qualified number restrictions
  - Very large KBs
  - Reasoning with individuals

Research Challenges

- **Increased expressive power**
  - Datatypes
  - Nominals
  - Extensions to DAML+OIL
- **Performance**
  - Inverse roles and qualified number restrictions
  - Very large KBs
  - Reasoning with individuals
- **Tools and Infrastructure**
  - Support for large scale ontological engineering and deployment

Research Challenges

- **Increased expressive power**
  - Datatypes
  - Nominals
  - Extensions to DAML+OIL
- **Performance**
  - Inverse roles and qualified number restrictions
  - Very large KBs
  - Reasoning with individuals
- **Tools and Infrastructure**
  - Support for large scale ontological engineering and deployment
- **New reasoning tasks**
  - Querying
  - Lcs/matching
  - . . .

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Datatypes

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Datatypes

☞ DAML+OIL has simple form of datatypes

- Unary predicates plus disjoint abstract/datatype domains

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Datatypes

- ☞ DAML+OIL has simple form of datatypes
  - Unary predicates plus disjoint abstract/datatype domains
- ☞ **Theoretically** not particularly challenging
  - Existing work on concrete domains [Baader & Hanschke, Lutz]
  - Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Datatypes

- ☞ DAML+OIL has simple form of datatypes
  - Unary predicates plus disjoint abstract/datatype domains
- ☞ **Theoretically** not particularly challenging
  - Existing work on concrete domains [Baader & Hanschke, Lutz]
  - Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]
- ☞ May be **practically** challenging
  - All XMLS datatypes supported

Increased Expressive Power: Datatypes

DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Datatypes

☞ DAML+OIL has simple form of datatypes

- Unary predicates plus disjoint abstract/datatype domains

☞ **Theoretically** not particularly challenging

- Existing work on concrete domains [Baader & Hanschke, Lutz]
- Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]

☞ May be **practically** challenging

- All XMLS datatypes supported

☞ Already seeing some (limited) **implementations**

- E.g., Cerebra system (Network Inference)

Increased Expressive Power: Nominals

Nominals

Increased Expressive Power: Nominals

Nominals

- DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic

Increased Expressive Power: Nominals

Nominals

- ☞ DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic
- ☞ Theoretically **very challenging**

Increased Expressive Power: Nominals

Nominals

- DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic
- Theoretically **very challenging**
  - Resulting logic has known high complexity (NExpTime)

Increased Expressive Power: Nominals

Nominals

- ☞ DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic
- ☞ Theoretically **very challenging**
  - Resulting logic has known high complexity (NExpTime)
  - No known “practical” algorithm

Increased Expressive Power: Nominals

Nominals

- DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic
- Theoretically **very challenging**
  - Resulting logic has known high complexity (NExpTime)
  - No known “practical” algorithm
  - Not obvious how to extend tableaux techniques in this direction
    - Loss of tree model property
    - Spy-points: $\top \sqsubseteq \exists R.\{Spy\}$
    - Finite domains: $\{Spy\} \sqsubseteq \leqslant nR^{-}$

Increased Expressive Power: Nominals

Nominals

- DAML+OIL has **oneOf** constructor
  - Extensionally defined concepts, e.g., $\{Mary\}^{\mathcal{I}} = \{Mary^{\mathcal{I}}\}$
  - Equivalent to nominals in modal logic
- Theoretically **very challenging**
  - Resulting logic has known high complexity (NExpTime)
  - No known “practical” algorithm
  - Not obvious how to extend tableaux techniques in this direction
    - → Loss of tree model property
    - → Spy-points: $\top \sqsubseteq \exists R.\{Spy\}$
    - → Finite domains: $\{Spy\} \sqsubseteq \leqslant nR^{-}$
- Relatively straightforward (in theory) without **inverse roles**
  - Algorithm for $\mathcal{SHOQ}(\mathbf{D})$ deals with nominals
  - Practical implementation still to be demonstrated

Increased Expressive Power: Extensions

☞ DAML+OIL **not expressive enough** for all applications

Increased Expressive Power: Extensions

- DAML+OIL **not expressive enough** for all applications
- Extensions **wish list** includes:
  - Complex roles/role inclusions, e.g., parent $\circ$ brother $\equiv$ uncle
  - Rules and/or query languages
  - Temporal and spatial reasoning
  - Defaults
  - . . .

Increased Expressive Power: Extensions

- DAML+OIL **not expressive enough** for all applications
- Extensions **wish list** includes:
  - Complex roles/role inclusions, e.g., parent $\circ$ brother $\equiv$ uncle
  - Rules and/or query languages
  - Temporal and spatial reasoning
  - Defaults
  - $\ldots$
- Extended language sure to be **undecidable**

Increased Expressive Power: Extensions

- DAML+OIL **not expressive enough** for all applications
- Extensions **wish list** includes:
  - Complex roles/role inclusions, e.g., parent $\circ$ brother $\equiv$ uncle
  - Rules and/or query languages
  - Temporal and spatial reasoning
  - Defaults
  - ...
- Extended language sure to be **undecidable**
- How can extensions best be **integrated** with DAML+OIL?

Increased Expressive Power: Extensions

- DAML+OIL **not expressive enough** for all applications
- Extensions **wish list** includes:
  - Complex roles/role inclusions, e.g., parent $\circ$ brother $\equiv$ uncle
  - Rules and/or query languages
  - Temporal and spatial reasoning
  - Defaults
  - ...
- Extended language sure to be **undecidable**
- How can extensions best be **integrated** with DAML+OIL?
- How can reasoners be developed/adapted for extended languages?

Performance Problems

- Evidence of **empirical tractability** mostly w.r.t. $\mathcal{SHF}$— problems can arise when systems extended to $\mathcal{SHIQ}$

Performance Problems

- ☞ Evidence of **empirical tractability** mostly w.r.t. $\mathcal{SHF}$— problems can arise when systems extended to $\mathcal{SHIQ}$
- ☞ Important **optimisations** no longer (fully) work
  - E.g., problems with caching as cached models can affect parent

Performance Problems

- ☞ Evidence of **empirical tractability** mostly w.r.t. $\mathcal{SHF}$— problems can arise when systems extended to $\mathcal{SHIQ}$
- ☞ Important **optimisations** no longer (fully) work
  - E.g., problems with caching as cached models can affect parent
- ☞ **Qualified number restrictions** can also cause problems
  - Even relatively small numbers can mean significant non-determinism

Performance Problems

- Evidence of **empirical tractability** mostly w.r.t. $\mathcal{SHF}$— problems can arise when systems extended to $\mathcal{SHIQ}$
- Important **optimisations** no longer (fully) work
  - E.g., problems with caching as cached models can affect parent
- **Qualified number restrictions** can also cause problems
  - Even relatively small numbers can mean significant non-determinism
- Reasoning with **very large KBs/ontologies**
  - Web ontologies can be expected to grow very large

Performance Problems

- Evidence of **empirical tractability** mostly w.r.t. $\mathcal{SHF}$— problems can arise when systems extended to $\mathcal{SHIQ}$
- Important **optimisations** no longer (fully) work
  - E.g., problems with caching as cached models can affect parent
- **Qualified number restrictions** can also cause problems
  - Even relatively small numbers can mean significant non-determinism
- Reasoning with **very large KBs/ontologies**
  - Web ontologies can be expected to grow very large
- Reasoning with **individuals** (Abox)
  - Deployment of web ontologies will mean reasoning with (possibly very large numbers of) individuals
  - Standard Abox techniques may not be able to cope

Performance Solutions (Maybe)

Performance Solutions (Maybe)

- Excessive **memory usage**

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**
  - Problem exacerbated by naive expansion rules
  - Promising results from optimised expansion using Algebraic Methods [Haarslev & Möller]

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**
  - Problem exacerbated by naive expansion rules
  - Promising results from optimised expansion using Algebraic Methods [Haarslev & Möller]
- **Caching** and merging

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**
  - Problem exacerbated by naive expansion rules
  - Promising results from optimised expansion using Algebraic Methods [Haarslev & Möller]
- **Caching** and merging
  - Can still work in some situations (work in progress)

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**
  - Problem exacerbated by naive expansion rules
  - Promising results from optimised expansion using Algebraic Methods [Haarslev & Möller]
- **Caching** and merging
  - Can still work in some situations (work in progress)
- Reasoning with **very large KBs**

Performance Solutions (Maybe)

- Excessive **memory usage**
  - Problem exacerbated by over-cautious double blocking condition (e.g., root node can never block)
  - Promising results from more precise blocking condition [Sattler & Horrocks]
- **Qualified number restrictions**
  - Problem exacerbated by naive expansion rules
  - Promising results from optimised expansion using Algebraic Methods [Haarslev & Möller]
- **Caching** and merging
  - Can still work in some situations (work in progress)
- Reasoning with **very large KBs**
  - DL systems shown to work with $\approx$100k concept KB [Haarslev & Möller]
  - But KB only exploited small part of DL language

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

☞ Ontology **design and maintenance**

- Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
- Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- ☞ Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- ☞ Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

☞ Ontology **design and maintenance**

- Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
- Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .

☞ Ontology **Integration**

- Some tools available, e.g., Chimera (Stanford)
- Need integrated **environments** . . .

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)
  - Need integrated **environments** . . .
  - Can learn from DB integration work [Lenzerini, Calvanese et al]

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)
  - Need integrated **environments** . . .
  - Can learn from DB integration work [Lenzerini, Calvanese et al]
- **Reasoning** engines

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)
  - Need integrated **environments** . . .
  - Can learn from DB integration work [Lenzerini, Calvanese et al]
- **Reasoning** engines
  - Several DL systems available

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

☞ Ontology **design and maintenance**

- Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
- Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .

☞ Ontology **Integration**

- Some tools available, e.g., Chimera (Stanford)
- Need integrated **environments** . . .
- Can learn from DB integration work [Lenzerini, Calvanese et al]

☞ **Reasoning** engines

- Several DL systems available
- Need for improved usability/connectivity

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)
  - Need integrated **environments** . . .
  - Can learn from DB integration work [Lenzerini, Calvanese et al]
- **Reasoning** engines
  - Several DL systems available
  - Need for improved usability/connectivity
  - DIG group recently formed for this purpose (and others)

Tools and Infrastructure

Tools and infrastructure required in order support use of DAML+OIL

- Ontology **design and maintenance**
  - Several **editors** available, e.g, OilEd (Manchester), OntoEdit (Karlsruhe), Protégé (Stanford)
  - Need integrated **environments** including modularity, versioning, visualisation, explanation, high-level languages, . . .
- Ontology **Integration**
  - Some tools available, e.g., Chimera (Stanford)
  - Need integrated **environments** . . .
  - Can learn from DB integration work [Lenzerini, Calvanese et al]
- **Reasoning** engines
  - Several DL systems available
  - Need for improved usability/connectivity
  - DIG group recently formed for this purpose (and others)
- . . .

Summary

Summary

☞ **Ontologies** will play key role in **Semantic Web**

Summary

- **Ontologies** will play key role in **Semantic Web**
- **DAML+OIL** is web ontology language based on **Description Logic**

Summary

- **Ontologies** will play key role in **Semantic Web**
- **DAML+OIL** is web ontology language based on **Description Logic**
- Ontology design, integration and deployment **supported by reasoning**

Summary

- **Ontologies** will play key role in **Semantic Web**
- **DAML+OIL** is web ontology language based on **Description Logic**
- Ontology design, integration and deployment **supported by reasoning**
- DLs are **logic based KR formalisms** with emphasis on reasoning

Summary

- **Ontologies** will play key role in **Semantic Web**
- **DAML+OIL** is web ontology language based on **Description Logic**
- Ontology design, integration and deployment **supported by reasoning**
- DLs are **logic based KR formalisms** with emphasis on reasoning
- DL systems provide **efficient reasoning services**
  - Careful choice of logic/algorithm
  - Highly optimised implementation

Resources

Slides from this talk

`www.cs.man.ac.uk/~horrocks/Slides/dagstuhl070202.pdf`

FaCT system

`www.cs.man.ac.uk/fact`

OIL

`www.ontoknowledge.org/oil/`

DAML+OIL

`www.daml.org/language/`

OilEd

`img.cs.man.ac.uk/oil`

I.COM

`www.cs.man.ac.uk/~franconi/icom/`

Select Bibliography

F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In B. Nebel, C. Rich, and W. Swartout, editors, *Proc. of KR'92*, pages 270–281. Morgan Kaufmann, 1992.

F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In *Proc. of KR'96*, pages 304–314. Morgan Kaufmann, 1996.

V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of IJCAI 2001* (to appear).

B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. In *Proc. of ECAI'90*, pages 348–353. John Wiley & Sons Ltd., 1990.

Select Bibliography

I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. In *Proc. of DL'98*, pages 90–94. CEUR, 1998.

I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of KR'00* pages 285–296. Morgan Kaufmann, 2000.

E. Franconi and G. Ng. The i.com tool for intelligent conceptual modelling. In *Proc. of (KRDB'00)*, August 2000.

D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

A. Levy and M.-C. Rousset". CARIN: A Representation Language Combining Horn Rules and Description Logics In *Proc. of (ECAI'96)*, 1996.