Reasoning with Expressive Description Logics

Logical Foundations for the Semantic Web

Ian Horrocks

horrocks@cs.man.ac.uk

University of Manchester

Manchester, UK

Talk Outline

Talk Outline

Introduction to Description Logics

Talk Outline

- Introduction to Description Logics
- The Semantic Web: Killer App for (DL) Reasoning?
  - Web Ontology Languages
  - DAML+OIL Language

Talk Outline

- Introduction to Description Logics
- The Semantic Web: Killer App for (DL) Reasoning?
    - Web Ontology Languages
    - DAML+OIL Language
- Reasoning with DAML+OIL
    - OilEd Demo

Talk Outline

- Introduction to Description Logics
- The Semantic Web: Killer App for (DL) Reasoning?
  - Web Ontology Languages
  - DAML+OIL Language
- Reasoning with DAML+OIL
  - OilEd Demo
- Description Logic Reasoning

Talk Outline

- Introduction to Description Logics
- The Semantic Web: Killer App for (DL) Reasoning?
  - Web Ontology Languages
  - DAML+OIL Language
- Reasoning with DAML+OIL
  - OilEd Demo
- Description Logic Reasoning
- Research Challenges

Introduction to Description Logics

What are Description Logics?

What are Description Logics?

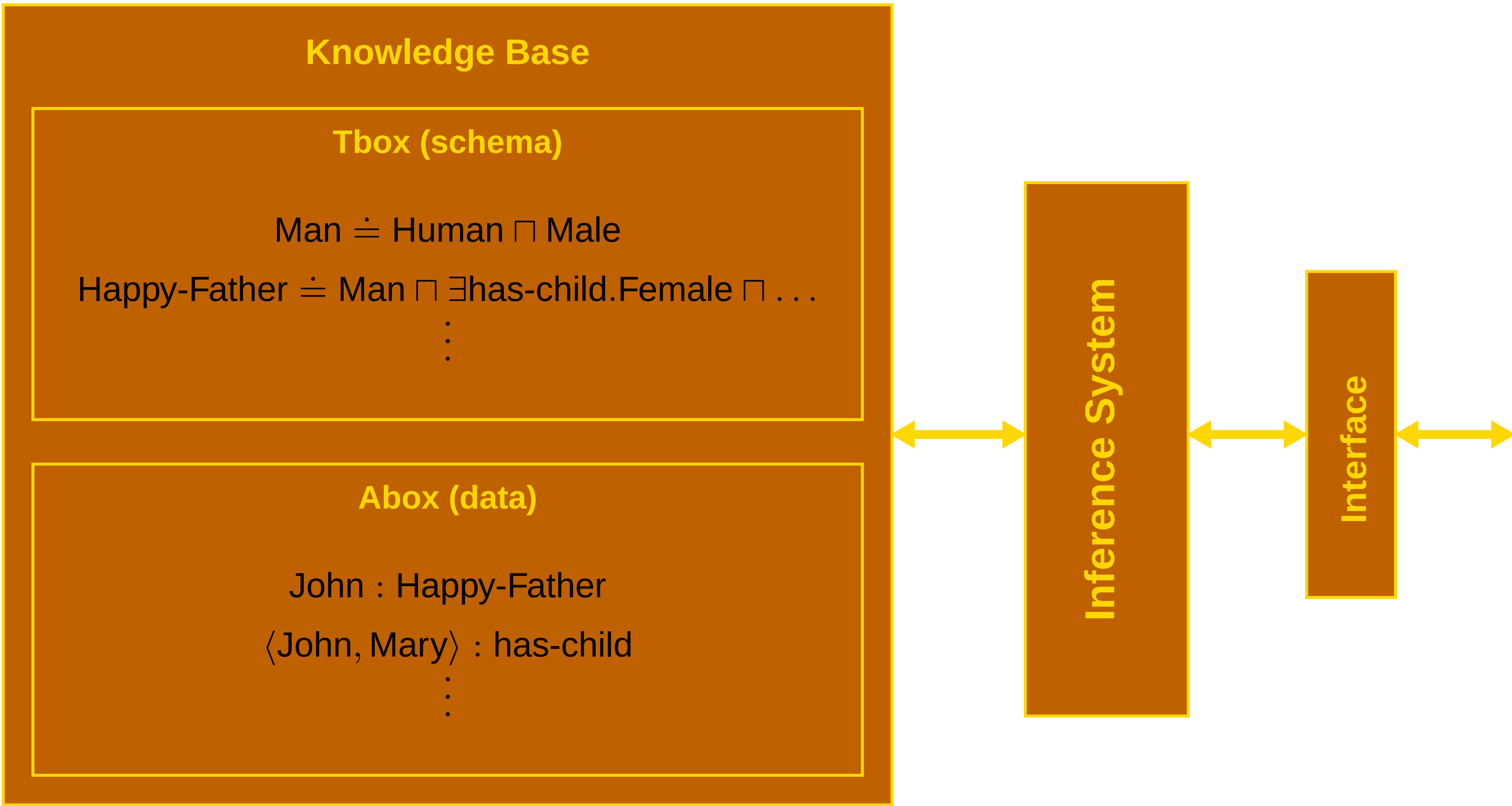
☞ A family of logic based Knowledge Representation formalisms

- Descendants of **semantic networks** and **KL-ONE**
- Describe domain in terms of **concepts** (classes), **roles** (relationships) and **individuals**

What are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of **semantic networks** and **KL-ONE**
  - Describe domain in terms of **concepts** (classes), **roles** (relationships) and **individuals**
- Distinguished by:
  - **Formal semantics** (model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of **inference services**
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)

DL Architecture



Short History of Description Logics

Short History of Description Logics

Phase 1:

- ☞ Incomplete systems (Back, Classic, Loom, . . .)
- ☞ Based on **structural algorithms**

Short History of Description Logics

Phase 1:

- ☞ Incomplete systems (Back, Classic, Loom, . . .)
- ☞ Based on **structural algorithms**

Phase 2:

- ☞ Development of **tableau algorithms** and **complexity results**
- ☞ Tableau-based systems for **Pspace** logics (e.g., Kris, Crack)
- ☞ Investigation of optimisation techniques

Short History of Description Logics

Phase 1:

- Incomplete systems (Back, Classic, Loom, . . .)
- Based on **structural algorithms**

Phase 2:

- Development of **tableau algorithms** and **complexity results**
- Tableau-based systems for **Pspace** logics (e.g., Kris, Crack)
- Investigation of optimisation techniques

Phase 3:

- Tableau algorithms for **very expressive** DLs
- **Highly optimised** tableau systems for **ExpTime** logics (e.g., FaCT, DLP, Racer)
- Relationship to modal logic and decidable fragments of FOL

Latest Developments

Phase 4:

Latest Developments

Phase 4:

- ☞ Mature **implementations**

Latest Developments

Phase 4:

- ☞ Mature **implementations**
- ☞ Mainstream **applications** and Tools
  - **Databases**
    - Consistency of conceptual schemata (EER, UML etc.)
    - Schema integration
    - Query subsumption (w.r.t. a conceptual schema)
  - Ontologies and **Semantic Web** (and **Grid**)
    - Ontology engineering (design, maintenance, integration)
    - Reasoning with ontology-based markup (meta-data)
    - Service description and discovery

Latest Developments

Phase 4:

- ☞ Mature **implementations**
- ☞ Mainstream **applications** and Tools
  - **Databases**
    - – Consistency of conceptual schemata (EER, UML etc.)
    - – Schema integration
    - – Query subsumption (w.r.t. a conceptual schema)
  - Ontologies and **Semantic Web** (and **Grid**)
    - – Ontology engineering (design, maintenance, integration)
    - – Reasoning with ontology-based markup (meta-data)
    - – Service description and discovery
- ☞ **Commercial** implementations
  - Cerebra system from Network Inference Ltd

The Semantic Web

The Semantic Web Vision

The Semantic Web Vision

☞ Web made possible through established **standards**

- **TCP/IP** for transporting bits down a wire
- **HTTP & HTML** for transporting and rendering hyperlinked text

The Semantic Web Vision

- ☞ Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- ☞ **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active
- Both intended for direct human processing/interaction

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active
- Both intended for direct human processing/interaction
- In **next generation** web, **resources** should be more accessible to automated processes

The Semantic Web Vision

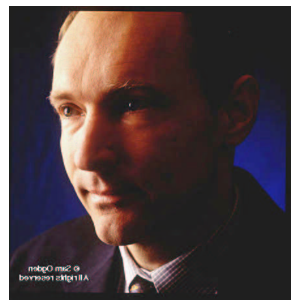
- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active
- Both intended for direct human processing/interaction
- In **next generation** web, **resources** should be more accessible to automated processes
  - To be achieved via **semantic markup**
  - **Metadata** annotations that describe content/function

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active
- Both intended for direct human processing/interaction
- In **next generation** web, **resources** should be more accessible to automated processes
  - To be achieved via **semantic markup**
  - **Metadata** annotations that describe content/function
- Coincides with Tim Berners-Lee's vision of a **Semantic Web**

The Semantic Web Vision

- Web made possible through established **standards**
  - **TCP/IP** for transporting bits down a wire
  - **HTTP & HTML** for transporting and rendering hyperlinked text
- **Applications** able to exploit this common infrastructure
  - Result is the WWW as we know it
- **1st generation** web mostly handwritten HTML pages
- **2nd generation** (current) web often machine generated/active
- Both intended for direct human processing/interaction
- In **next generation** web, **resources** should be more accessible to automated processes
  - To be achieved via **semantic markup**
  - **Metadata** annotations that describe content/function
- Coincides with Tim Berners-Lee's vision of a **Semantic Web**



Ontologies

Ontologies

☞ Semantic markup must be **meaningful** to automated processes

Ontologies

- ☞ Semantic markup must be **meaningful** to automated processes
- ☞ Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)

Ontologies

- ☞ Semantic markup must be **meaningful** to automated processes
- ☞ Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- ☞ Ontology typically consists of:
  - **Hierarchical** description of important **concepts** in domain
  - Descriptions of **properties** of instances of each concept

Ontologies

- ☞ Semantic markup must be **meaningful** to automated processes
- ☞ Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- ☞ Ontology typically consists of:
  - **Hierarchical** description of important **concepts** in domain
  - Descriptions of **properties** of instances of each concept
- ☞ Degree of formality can be quite variable (NL–logic)

Ontologies

- Semantic markup must be **meaningful** to automated processes
- Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- Ontology typically consists of:
  - **Hierarchical** description of important **concepts** in domain
  - Descriptions of **properties** of instances of each concept
- Degree of formality can be quite variable (NL–logic)
- Increased formality and regularity facilitates machine understanding

Ontologies

- Semantic markup must be **meaningful** to automated processes
- Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- Ontology typically consists of:
  - **Hierarchical** description of important **concepts** in domain
  - Descriptions of **properties** of instances of each concept
- Degree of formality can be quite variable (NL–logic)
- Increased formality and regularity facilitates machine understanding
- Ontologies can be used, e.g.:
  - To facilitate buyer–seller communication in **e-commerce**
  - In semantic based **search**
  - To provide richer **service descriptions** that can be more flexibly interpreted by intelligent agents

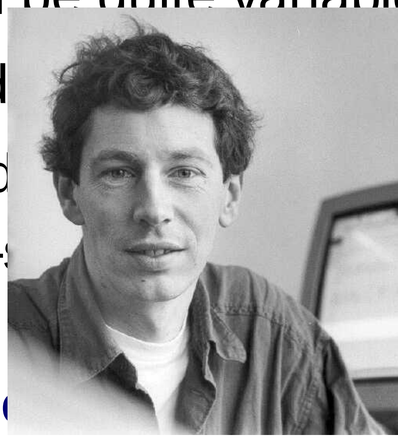
Ontologies

- Semantic markup must be **meaningful** to automated processes
- Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- Ontology typically consists of:

  - scription of important **concepts** in domain
  - **roperties** of instances of each concept
- Degr an be quite variable (NL–logic)
- Incre nd regularity facilitates machine understanding
- Onto sed, e.g.:
  - er–seller communication in **e-commerce**
  - ed **search**
  - r **service descriptions** that can be more flexibly interpreted by intelligent agents

Ontologies

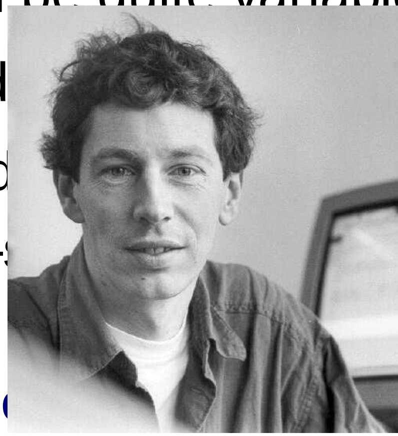
- Semantic markup must be **meaningful** to automated processes
- Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- Ontology typically consists of:



  - scription of important **concepts** in domain
  - **roperties** of instances of each concept
- Degr an be quite variable (NL–logic)
- Incre nd ates machine understanding
- Onto sed
  - er– ation in **e-commerce**
  - ed
  - r **s** **ions** that can be more flexibly interpreted by intelligent agents

Ontologies

- Semantic markup must be **meaningful** to automated processes
- Ontologies will play a key role
  - Source of **precisely defined** terms (vocabulary)
  - Can be **shared** across applications (and humans)
- Ontology typically consists of:
  - scription of important **concepts** in domain
  - **roperties** of instanc ept
- Degr an be quite variable (I
- Incre nd ate rstanding
- Onto sed
  - er– ati **ce**
  - ed
  - r **s** **ion** ore flexibly interpreted by intelligent agents



Web Ontology Languages

Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects

Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects
- Efforts merged to produce **DAML+OIL**

Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects
- Efforts merged to produce **DAML+OIL**
  - Submitted to **W3C** as basis for **standardisation**
  - **WebOnt** working group developing **OWL** language standard

Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects
- Efforts merged to produce **DAML+OIL**
  - Submitted to **W3C** as basis for **standardisation**
  - **WebOnt** working group developing **OWL** language standard
- DAML+OIL/OWL **"layered"** on top of RDFS
  - RDFS based **syntax** and ontological primitives (subclass etc.)
  - Adds **much** richer set of primitives (transitivity, cardinality, . . .)

Web Ontology Languages

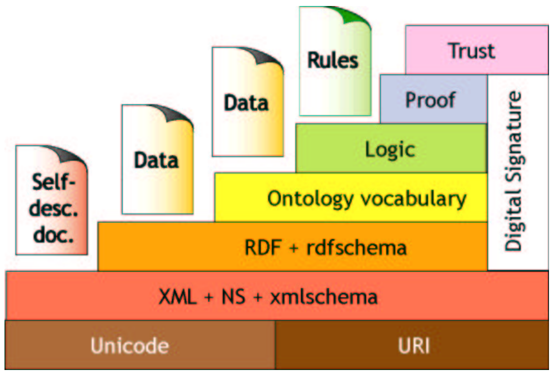
☞ **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects

☞ Efforts merged to produce **DAML+OIL**

- Submitted to **W3C** as basis for **standardisation**
- **WebOnt** working group developing **OWL** language standard

☞ DAML+OIL/OWL **"layered"** on top of RDFS

- RDFS based **syntax** and ontological primitives (subclass etc.)
- Adds **much** richer set of primitives (transitivity, cardinality, . . .)



Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects
- Efforts merged to produce **DAML+OIL**
  - Submitted to **W3C** as basis for **standardisation**
  - **WebOnt** working group developing **OWL** language standard
- DAML+OIL/OWL **“layered”** on top of RDFS
  - RDFS based **syntax** and ontological primitives (subclass etc.)
  - Adds **much** richer set of primitives (transitivity, cardinality, . . .)
- Describes class/property **structure** of domain (Tbox)
  - E.g., Person **subclass of** Animal whose parents are all Persons

Web Ontology Languages

- **OIL** and **DAML-ONT** web ontology languages developed in European and DARPA projects
- Efforts merged to produce **DAML+OIL**
  - Submitted to **W3C** as basis for **standardisation**
  - **WebOnt** working group developing **OWL** language standard
- DAML+OIL/OWL **"layered"** on top of RDFS
  - RDFS based **syntax** and ontological primitives (subclass etc.)
  - Adds **much** richer set of primitives (transitivity, cardinality, . . .)
- Describes class/property **structure** of domain (Tbox)
  - E.g., Person **subclass of** Animal whose parents are all Persons
- Uses RDF for class/property membership assertions (Abox)
  - E.g., john **instance of** Person; $\langle$john, mary$\rangle$ instance of parent

Logical Foundations of DAML+OIL

Logical Foundations of DAML+OIL

- DAML+OIL equivalent to very expressive **Description Logic**

Logical Foundations of DAML+OIL

- DAML+OIL equivalent to very expressive **Description Logic**
- More precisely, DAML+OIL is (extension of) $\mathcal{SHIQ}$ DL

Logical Foundations of DAML+OIL

- DAML+OIL equivalent to very expressive **Description Logic**
- More precisely, DAML+OIL is (extension of) $\mathcal{SHIQ}$ DL
- DAML+OIL benefits from many years of DL research
  - Well defined **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning algorithms**
  - **Implemented systems** (highly optimised)

Logical Foundations of DAML+OIL

- DAML+OIL equivalent to very expressive **Description Logic**
- More precisely, DAML+OIL is (extension of) $\mathcal{SHIQ}$ DL
- DAML+OIL benefits from many years of DL research
  - Well defined **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning algorithms**
  - **Implemented systems** (highly optimised)
- DAML+OIL classes can be names (URI's) or **expressions**
  - Various **constructors** provided for building class expressions

Logical Foundations of DAML+OIL

- DAML+OIL equivalent to very expressive **Description Logic**
- More precisely, DAML+OIL is (extension of) $\mathcal{SHIQ}$ DL
- DAML+OIL benefits from many years of DL research
  - Well defined **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning algorithms**
  - **Implemented systems** (highly optimised)
- DAML+OIL classes can be names (URI's) or **expressions**
  - Various **constructors** provided for building class expressions
- **Expressive power** determined by
  - Kinds of constructor provided
  - Kinds of axiom allowed

DAML+OIL Class Constructors

DAML+OIL Class Constructors

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}	$x_1 \vee \ldots \vee x_n$
toClass	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinalityQ	$\leqslant n P.C$	$\leqslant 1$hasChild.Male	$[P]_{n+1} C$
minCardinalityQ	$\geqslant n P.C$	$\geqslant 2$hasChild.Lawyer	$\langle P \rangle_n C$

DAML+OIL Class Constructors

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}	$x_1 \vee \ldots \vee x_n$
toClass	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinalityQ	$\leqslant n P.C$	$\leqslant 1$hasChild.Male	$[P]_{n+1}C$
minCardinalityQ	$\geqslant n P.C$	$\geqslant 2$hasChild.Lawyer	$\langle P \rangle_n C$

- ☞ XMLS **datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists$hasAge.nonNegativeInteger

DAML+OIL Class Constructors

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}	$x_1 \vee \ldots \vee x_n$
toClass	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinalityQ	$\leqslant n P.C$	$\leqslant$1hasChild.Male	$[P]_{n+1} C$
minCardinalityQ	$\geqslant n P.C$	$\geqslant$2hasChild.Lawyer	$\langle P \rangle_n C$

☞ XMLS **datatypes** as well as classes in $\forall P.C$ and $\exists P.C$

- E.g., $\exists$hasAge.nonNegativeInteger

☞ Arbitrarily complex **nesting** of constructors

- E.g., Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)

RDFS Syntax

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

Semantics

Semantics

☞ Semantics defined by **interpretations**: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- concepts $\longrightarrow$ subsets of $\Delta^{\mathcal{I}}$
- roles $\longrightarrow$ binary relations over $\Delta^{\mathcal{I}}$ (subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$)
- individuals $\longrightarrow$ elements of $\Delta^{\mathcal{I}}$

Semantics

☞ Semantics defined by **interpretations**: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- concepts $\longrightarrow$ subsets of $\Delta^{\mathcal{I}}$
- roles $\longrightarrow$ binary relations over $\Delta^{\mathcal{I}}$ (subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$)
- individuals $\longrightarrow$ elements of $\Delta^{\mathcal{I}}$

☞ Interpretation function $\cdot^{\mathcal{I}}$ **extended** to concept expressions

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \quad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $\{x_n, \ldots, x_n\}^{\mathcal{I}} = \{x_n^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
- $(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
- $(\leqslant nR.C)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leqslant n\}$
- $(\geqslant nR.C)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geqslant n\}$

DAML+OIL Axioms

DAML+OIL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$hasSSN$^-$

DAML+OIL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$hasSSN$^-$

☞ $\mathcal{I}$ **satisfies** $C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$; satisfies $P_1 \sqsubseteq P_2$ iff $P_1^{\mathcal{I}} \subseteq P_2^{\mathcal{I}}$

DAML+OIL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$hasSSN$^-$

- ☞ $\mathcal{I}$ **satisfies** $C_1 \sqsubseteq C_2$ iff $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$; satisfies $P_1 \sqsubseteq P_2$ iff $P_1^\mathcal{I} \subseteq P_2^\mathcal{I}$
- ☞ $\mathcal{I}$ satisfies ontology $\mathcal{O}$ (is a **model** of $\mathcal{O}$) iff satisfies every axiom in $\mathcal{O}$

XML Datatypes in DAML+OIL

XML Datatypes in DAML+OIL

- ☞ DAML+OIL supports **XML Schema** datatypes
  - Primitive (e.g., decimal) and derived (e.g., integer sub-range)

XML Datatypes in DAML+OIL

- DAML+OIL supports **XML Schema** datatypes
  - Primitive (e.g., decimal) and derived (e.g., integer sub-range)
- Clean **separation** between “object” classes and datatypes
  - Disjoint interpretation domain: $d^{\mathcal{I}} \subseteq \Delta_{\mathbf{D}}$, and $\Delta_{\mathbf{D}} \cap \Delta^{\mathcal{I}} = \emptyset$
  - Disjoint datatype properties: $P_{\mathbf{D}}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}$

XML Datatypes in DAML+OIL

- DAML+OIL supports **XML Schema** datatypes
  - Primitive (e.g., decimal) and derived (e.g., integer sub-range)
- Clean **separation** between “object” classes and datatypes
  - Disjoint interpretation domain: $d^{\mathcal{I}} \subseteq \Delta_{\mathbf{D}}$, and $\Delta_{\mathbf{D}} \cap \Delta^{\mathcal{I}} = \emptyset$
  - Disjoint datatype properties: $P_{\mathbf{D}}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}$
- Philosophical reasons:
  - Datatypes structured by **built-in predicates**
  - Not appropriate to form new datatypes using ontology language

XML Datatypes in DAML+OIL

- DAML+OIL supports **XML Schema** datatypes
  - Primitive (e.g., decimal) and derived (e.g., integer sub-range)
- Clean **separation** between “object” classes and datatypes
  - Disjoint interpretation domain: $d^{\mathcal{I}} \subseteq \Delta_{\mathbf{D}}$, and $\Delta_{\mathbf{D}} \cap \Delta^{\mathcal{I}} = \emptyset$
  - Disjoint datatype properties: $P_{\mathbf{D}}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}}$
- Philosophical reasons:
  - Datatypes structured by **built-in predicates**
  - Not appropriate to form new datatypes using ontology language
- Practical reasons:
  - Ontology language remains **simple and compact**
  - **Semantic integrity** of ontology language not compromised
  - **Implementability** not compromised — can use hybrid reasoner
    - Only need sound and complete decision procedure for $d_1^{\mathcal{I}} \cap \ldots \cap d_n^{\mathcal{I}}$, where d_i is a (possibly negated) datatype

Reasoning with DAML+OIL

Reasoning

Reasoning

☞ Why do we want it?

Reasoning

- Why do we want it?
  - Semantic Web aims at “machine understanding”
  - **Understanding** closely related to **reasoning**

Reasoning

- ☞ Why do we want it?
  - Semantic Web aims at “machine understanding”
  - **Understanding** closely related to **reasoning**
- ☞ What can we do with it?

Reasoning

- Why do we want it?
  - Semantic Web aims at “machine understanding”
  - **Understanding** closely related to **reasoning**
- What can we do with it?
  - **Design and maintenance** of ontologies
    - Check class consistency and compute class hierarchy
    - Particularly important with large ontologies/multiple authors

Reasoning

- Why do we want it?
  - Semantic Web aims at “machine understanding”
  - **Understanding** closely related to **reasoning**
- What can we do with it?
  - **Design and maintenance** of ontologies
    - Check class consistency and compute class hierarchy
    - Particularly important with large ontologies/multiple authors
  - **Integration** of ontologies
    - Assert inter-ontology relationships
    - Reasoner computes integrated class hierarchy/consistency

Reasoning

- Why do we want it?
  - Semantic Web aims at “machine understanding”
  - **Understanding** closely related to **reasoning**
- What can we do with it?
  - **Design and maintenance** of ontologies
    - Check class consistency and compute class hierarchy
    - Particularly important with large ontologies/multiple authors
  - **Integration** of ontologies
    - Assert inter-ontology relationships
    - Reasoner computes integrated class hierarchy/consistency
  - **Querying** class and instance data w.r.t. ontologies
    - Determine if set of facts are consistent w.r.t. ontologies
    - Determine if individuals are instances of ontology classes
    - Retrieve individuals/tuples satisfying a query expression
    - Check if one description more general than another w.r.t. ontology
    - . . .

Basic Inference Problems

Basic Inference Problems

☞ **Consistency** — check if knowledge is meaningful

- Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
- Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$

Basic Inference Problems

- **Consistency** — check if knowledge is meaningful
  - Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
  - Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$
- **Subsumption** — structure knowledge, compute taxonomy
  - $C \sqsubseteq_{\mathcal{O}} D$? $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$

Basic Inference Problems

- ☞ **Consistency** — check if knowledge is meaningful
  - Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
  - Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$
- ☞ **Subsumption** — structure knowledge, compute taxonomy
  - $C \sqsubseteq_{\mathcal{O}} D$? $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- ☞ **Equivalence** — check if two classes denote same set of instances
  - $C \equiv_{\mathcal{O}} D$? $C^{\mathcal{I}} = D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$

Basic Inference Problems

☞ **Consistency** — check if knowledge is meaningful

- Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
- Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$

☞ **Subsumption** — structure knowledge, compute taxonomy

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

☞ **Equivalence** — check if two classes denote same set of instances

- $C \equiv_{\mathcal{O}} D$? $C^{\mathcal{I}} = D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$

☞ **Instantiation** — check if individual i instance of class C

- $i \in_{\mathcal{O}} C$? $i \in C^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$

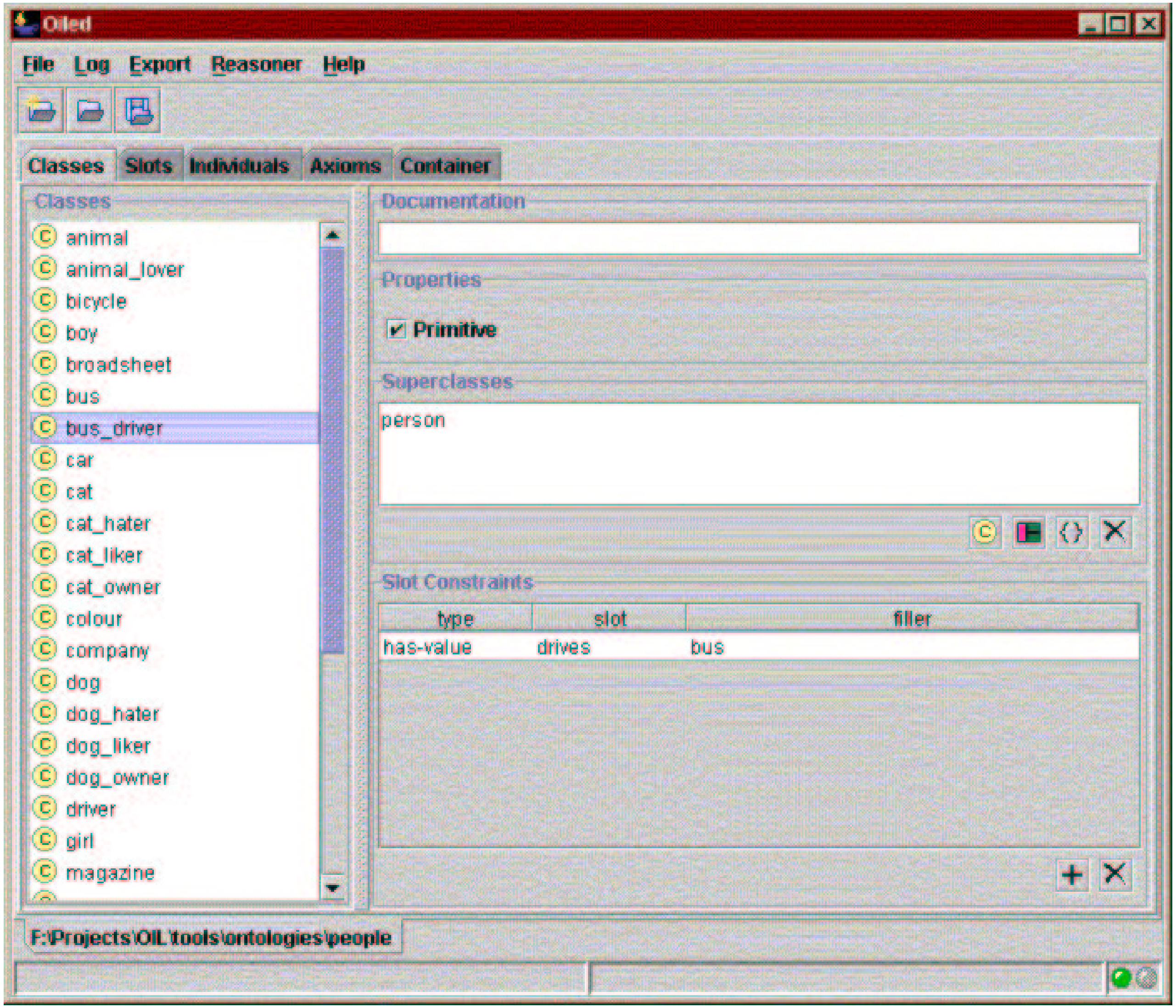
Basic Inference Problems

- **Consistency** — check if knowledge is meaningful
  - Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
  - Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$
- **Subsumption** — structure knowledge, compute taxonomy
  - $C \sqsubseteq_{\mathcal{O}} D$? $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Equivalence** — check if two classes denote same set of instances
  - $C \equiv_{\mathcal{O}} D$? $C^{\mathcal{I}} = D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Instantiation** — check if individual i instance of class C
  - $i \in_{\mathcal{O}} C$? $i \in C^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Retrieval** — retrieve set of individuals that instantiate C
  - set of i s.t. $i \in C^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$

Basic Inference Problems

- **Consistency** — check if knowledge is meaningful
  - Is $\mathcal{O}$ consistent? There exists some model $\mathcal{I}$ of $\mathcal{O}$
  - Is C consistent? $C^{\mathcal{I}} \neq \emptyset$ in some model $\mathcal{I}$ of $\mathcal{O}$
- **Subsumption** — structure knowledge, compute taxonomy
  - $C \sqsubseteq_{\mathcal{O}} D$? $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Equivalence** — check if two classes denote same set of instances
  - $C \equiv_{\mathcal{O}} D$? $C^{\mathcal{I}} = D^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Instantiation** — check if individual i instance of class C
  - $i \in_{\mathcal{O}} C$? $i \in C^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- **Retrieval** — retrieve set of individuals that instantiate C
  - set of i s.t. $i \in C^{\mathcal{I}}$ in all models $\mathcal{I}$ of $\mathcal{O}$
- Problems all **reducible** to consistency (satisfiability):
  - $C \sqsubseteq_{\mathcal{O}} D$ iff $C \sqcap \neg D$ not consistent w.r.t. $\mathcal{O}$
  - $i \in_{\mathcal{O}} C$ iff $\mathcal{O} \cup \{i \in \neg C\}$ is **not** consistent

Reasoning Support for Ontology Design: OilEd



Description Logic Reasoning

Tableaux Algorithms — Basics

Tableaux Algorithms — Basics

☞ Tableaux algorithms used to test **satisfiability**

Tableaux Algorithms — Basics

- ☞ Tableaux algorithms used to test **satisfiability**
- ☞ Try to build **tree-like model** $\mathcal{I}$ of input concept C

Tableaux Algorithms — Basics

- Tableaux algorithms used to test **satisfiability**
- Try to build **tree-like model** $\mathcal{I}$ of input concept C
- Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \rightsquigarrow \forall R.\neg C$ etc.

Tableaux Algorithms — Basics

- ☞ Tableaux algorithms used to test **satisfiability**
- ☞ Try to build **tree-like model** $\mathcal{I}$ of input concept C
- ☞ Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \rightsquigarrow \forall R.\neg C$ etc.
- ☞ Break down C **syntactically**, inferring constraints on elements of $\mathcal{I}$

Tableaux Algorithms — Basics

- Tableaux algorithms used to test **satisfiability**
- Try to build **tree-like model** $\mathcal{I}$ of input concept C
- Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \rightsquigarrow \forall R.\neg C$ etc.
- Break down C **syntactically**, inferring constraints on elements of $\mathcal{I}$
- Decomposition uses **tableau rules** corresponding to constructors in logic (e.g., $\sqcap$, $\exists$)
  - Some rules are **nondeterministic** (e.g., $\sqcup$, $\leqslant$)
  - In practice, this means **search**

Tableaux Algorithms — Basics

- Tableaux algorithms used to test **satisfiability**
- Try to build **tree-like model** $\mathcal{I}$ of input concept C
- Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \leadsto \forall R.\neg C$ etc.
- Break down C **syntactically**, inferring constraints on elements of $\mathcal{I}$
- Decomposition uses **tableau rules** corresponding to constructors in logic (e.g., $\sqcap$, $\exists$)
  - Some rules are **nondeterministic** (e.g., $\sqcup$, $\leqslant$)
  - In practice, this means **search**
- Stop when **clash** occurs or when no rules are applicable

Tableaux Algorithms — Basics

- Tableaux algorithms used to test **satisfiability**
- Try to build **tree-like model** $\mathcal{I}$ of input concept C
- Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \rightsquigarrow \forall R.\neg C$ etc.
- Break down C **syntactically**, inferring constraints on elements of $\mathcal{I}$
- Decomposition uses **tableau rules** corresponding to constructors in logic (e.g., $\sqcap$, $\exists$)
  - Some rules are **nondeterministic** (e.g., $\sqcup$, $\leqslant$)
  - In practice, this means **search**
- Stop when **clash** occurs or when no rules are applicable
- **Blocking** (cycle check) used to guarantee **termination**

Tableaux Algorithms — Basics

- Tableaux algorithms used to test **satisfiability**
- Try to build **tree-like model** $\mathcal{I}$ of input concept C
- Work on concepts in **negation normal form**
  - Push in negation using de Morgan's, $\neg\exists R.C \rightsquigarrow \forall R.\neg C$ etc.
- Break down C **syntactically**, inferring constraints on elements of $\mathcal{I}$
- Decomposition uses **tableau rules** corresponding to constructors in logic (e.g., $\sqcap$, $\exists$)
  - Some rules are **nondeterministic** (e.g., $\sqcup$, $\leqslant$)
  - In practice, this means **search**
- Stop when **clash** occurs or when no rules are applicable
- **Blocking** (cycle check) used to guarantee **termination**
- Return "C is consistent" **iff** C is consistent
  - Tree model property

Tableaux Algorithms — Details

Tableaux Algorithms — Details

☞ Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C

- Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
- Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$

Tableaux Algorithms — Details

- Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C
  - Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
  - Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- $\mathbf{T}$ initialised with single **root node** labeled $\{C\}$

Tableaux Algorithms — Details

- Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C
  - Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
  - Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- $\mathbf{T}$ initialised with single **root node** labeled $\{C\}$
- **Tableau rules** repeatedly applied to node labels
  - Extend labels or extend/modify $\mathbf{T}$ structure
  - Rules can be **blocked**, e.g, if predecessor has **superset** label
  - Nondeterministic rules $\longrightarrow$ **search** possible extensions

Tableaux Algorithms — Details

- Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C
  - Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
  - Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- $\mathbf{T}$ initialised with single **root node** labeled $\{C\}$
- **Tableau rules** repeatedly applied to node labels
  - Extend labels or extend/modify $\mathbf{T}$ structure
  - Rules can be **blocked**, e.g, if predecessor has **superset** label
  - Nondeterministic rules $\longrightarrow$ **search** possible extensions
- $\mathbf{T}$ contains **Clash** if obvious contradiction in some node label
  - E.g., $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some concept A and node x

Tableaux Algorithms — Details

- Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C
  - Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
  - Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- $\mathbf{T}$ initialised with single **root node** labeled $\{C\}$
- **Tableau rules** repeatedly applied to node labels
  - Extend labels or extend/modify $\mathbf{T}$ structure
  - Rules can be **blocked**, e.g, if predecessor has **superset** label
  - Nondeterministic rules $\longrightarrow$ **search** possible extensions
- $\mathbf{T}$ contains **Clash** if obvious contradiction in some node label
  - E.g., $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some concept A and node x
- $\mathbf{T}$ **fully expanded** if no rules are applicable

Tableaux Algorithms — Details

- Work on **tree** $\mathbf{T}$ representing **model** $\mathcal{I}$ of concept C
  - Nodes represent elements of $\Delta^{\mathcal{I}}$; labeled with subconcepts of C
  - Edges represent role-successorships between elements of $\Delta^{\mathcal{I}}$
- $\mathbf{T}$ initialised with single **root node** labeled $\{C\}$
- **Tableau rules** repeatedly applied to node labels
  - Extend labels or extend/modify $\mathbf{T}$ structure
  - Rules can be **blocked**, e.g, if predecessor has **superset** label
  - Nondeterministic rules $\longrightarrow$ **search** possible extensions
- $\mathbf{T}$ contains **Clash** if obvious contradiction in some node label
  - E.g., $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some concept A and node x
- $\mathbf{T}$ **fully expanded** if no rules are applicable
- C satisfiable iff fully expanded clash free $\mathbf{T}$ found
  - Trivial correspondence between such a $\mathbf{T}$ and a model of C

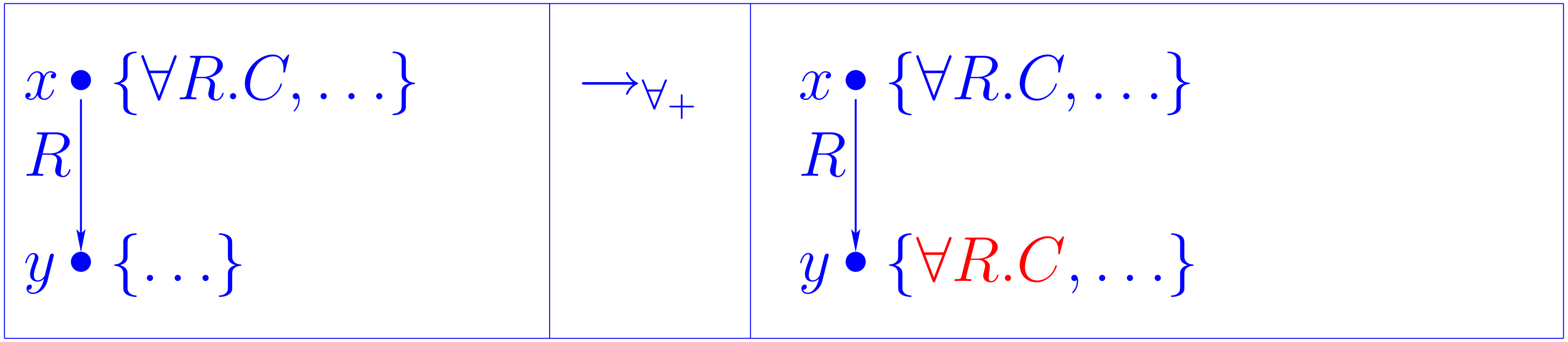
Tableaux Rules for $\mathcal{ALC}$

Tableaux Rules for $\mathcal{ALC}$

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

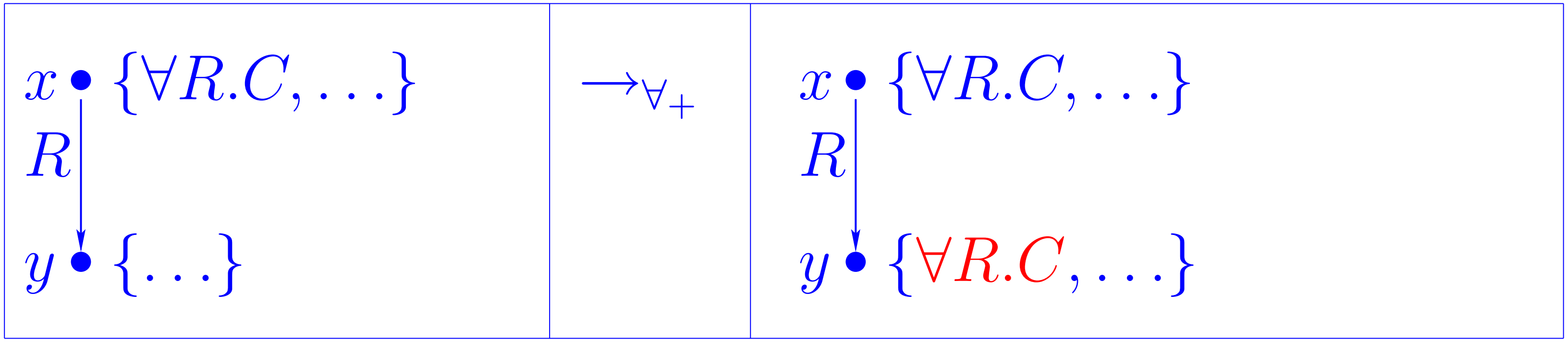
Tableaux Rule for Transitive Roles

Tableaux Rule for Transitive Roles



Where R is a transitive role (i.e., $(R^{\mathcal{I}})^+ = R^{\mathcal{I}}$)

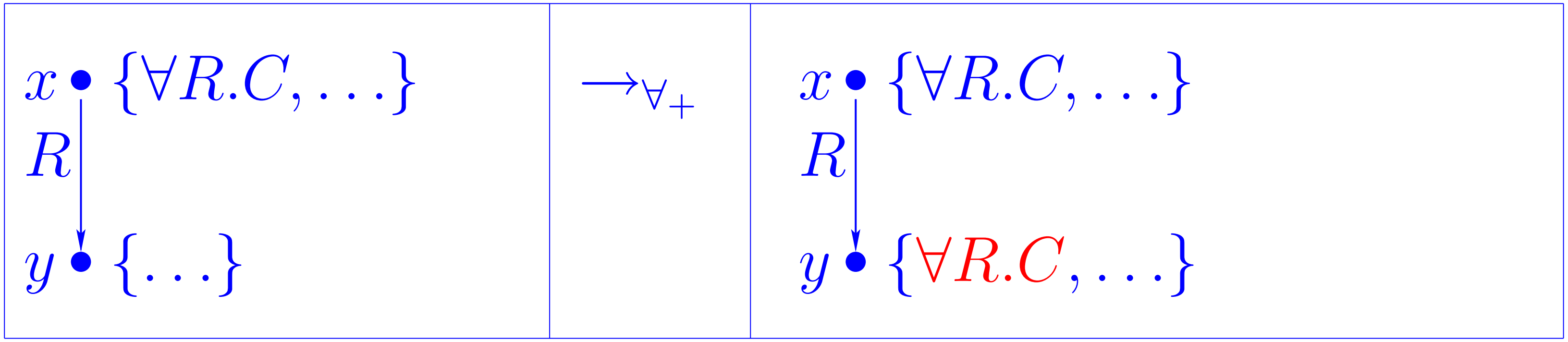
Tableaux Rule for Transitive Roles



Where R is a transitive role (i.e., $(R^{\mathcal{I}})^+ = R^{\mathcal{I}}$)

☞ No longer naturally terminating (e.g., if $C = \exists R.\top$)

Tableaux Rule for Transitive Roles



Where R is a transitive role (i.e., $(R^{\mathcal{I}})^{+} = R^{\mathcal{I}}$)

- No longer naturally terminating (e.g., if $C = \exists R.\top$)
- Need blocking
  - Simple blocking suffices for $\mathcal{ALC}$ plus transitive roles
  - I.e., do not expand node label if ancestor has superset label
  - More expressive logics (e.g., with inverse roles) need more sophisticated blocking strategies

Tableaux Algorithm — Example

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 Algorithm — Example

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

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

(w)

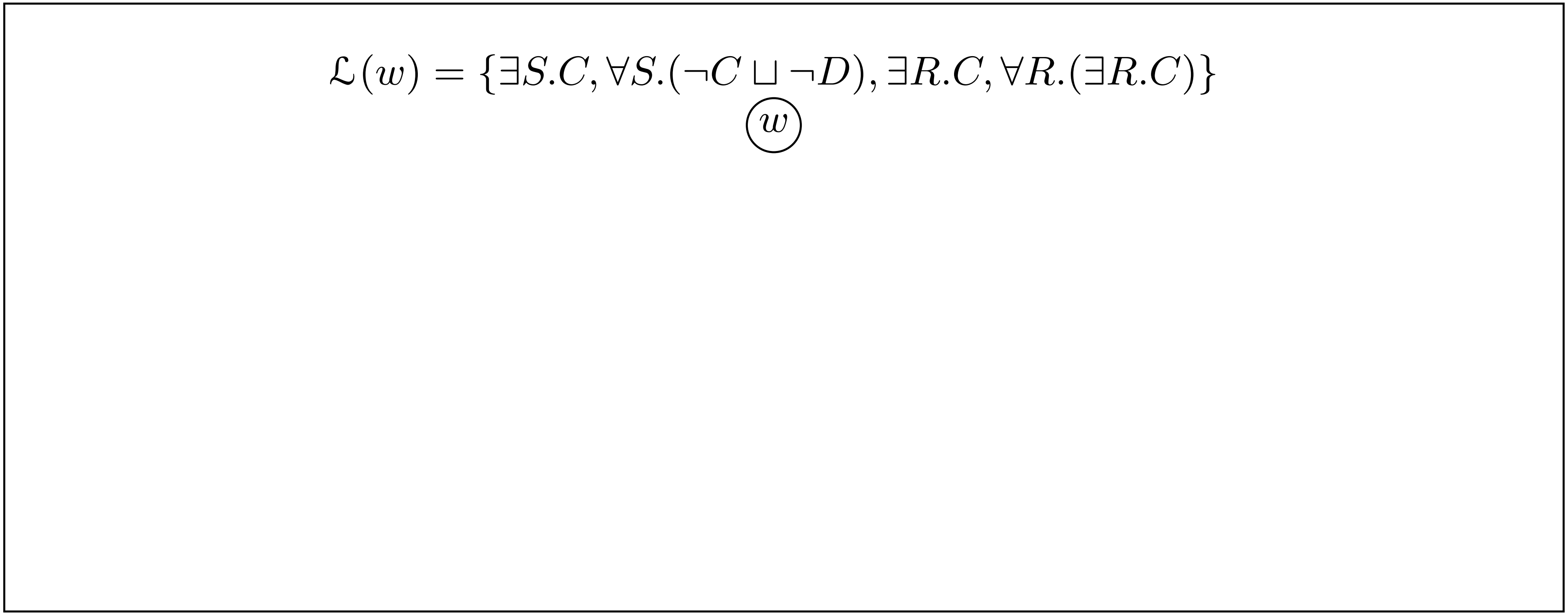
Tableaux Algorithm — Example

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

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

Tableaux Algorithm — Example

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



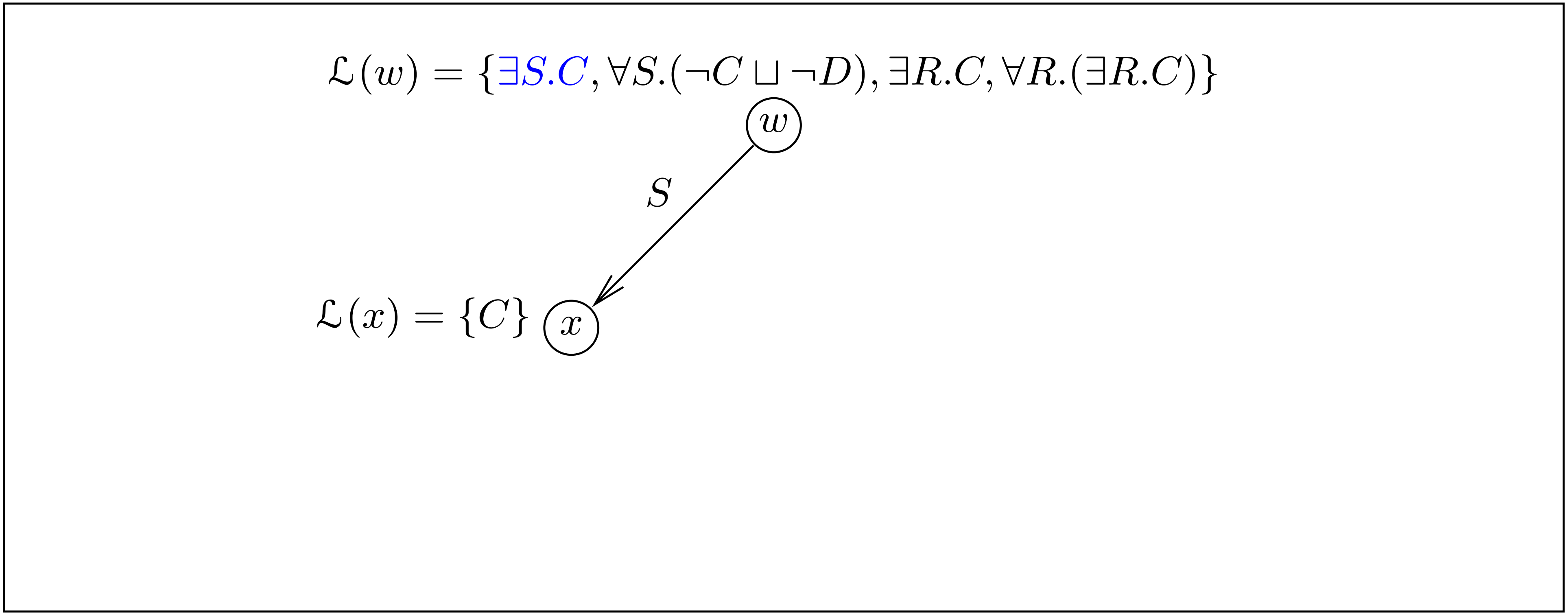
Tableaux Algorithm — Example

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

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

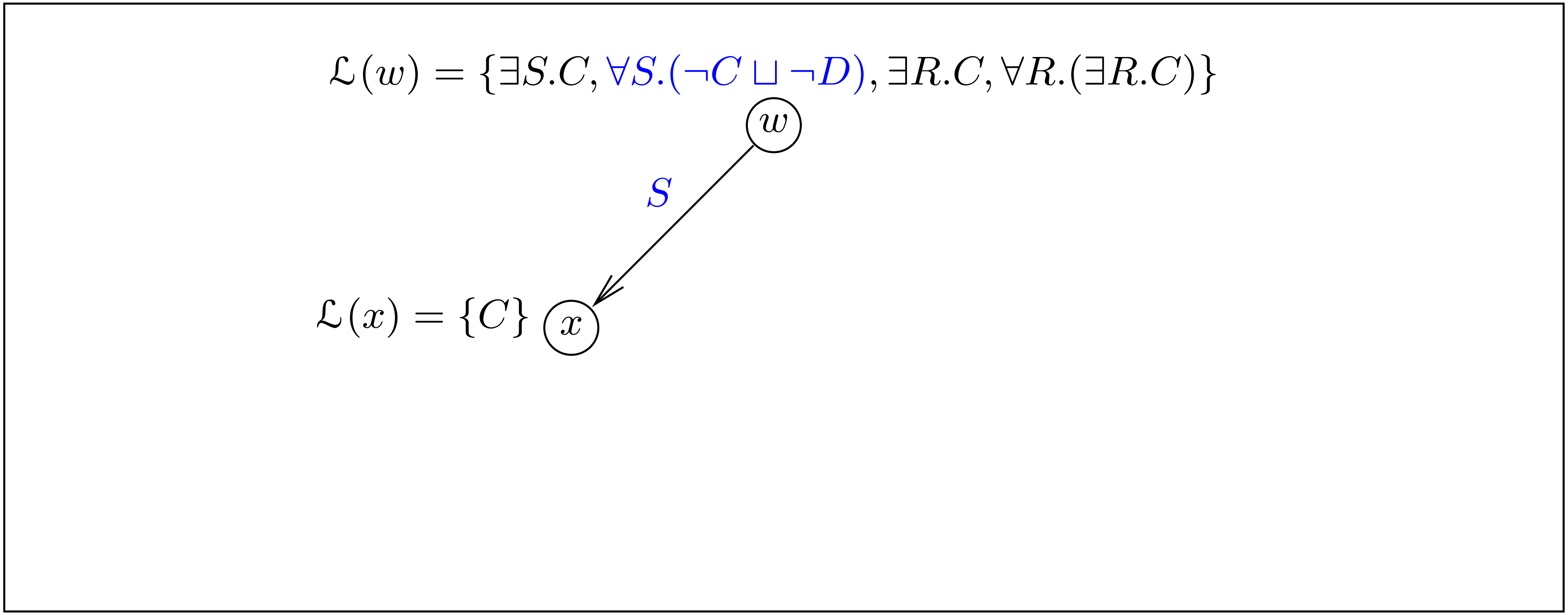
Tableaux Algorithm — Example

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



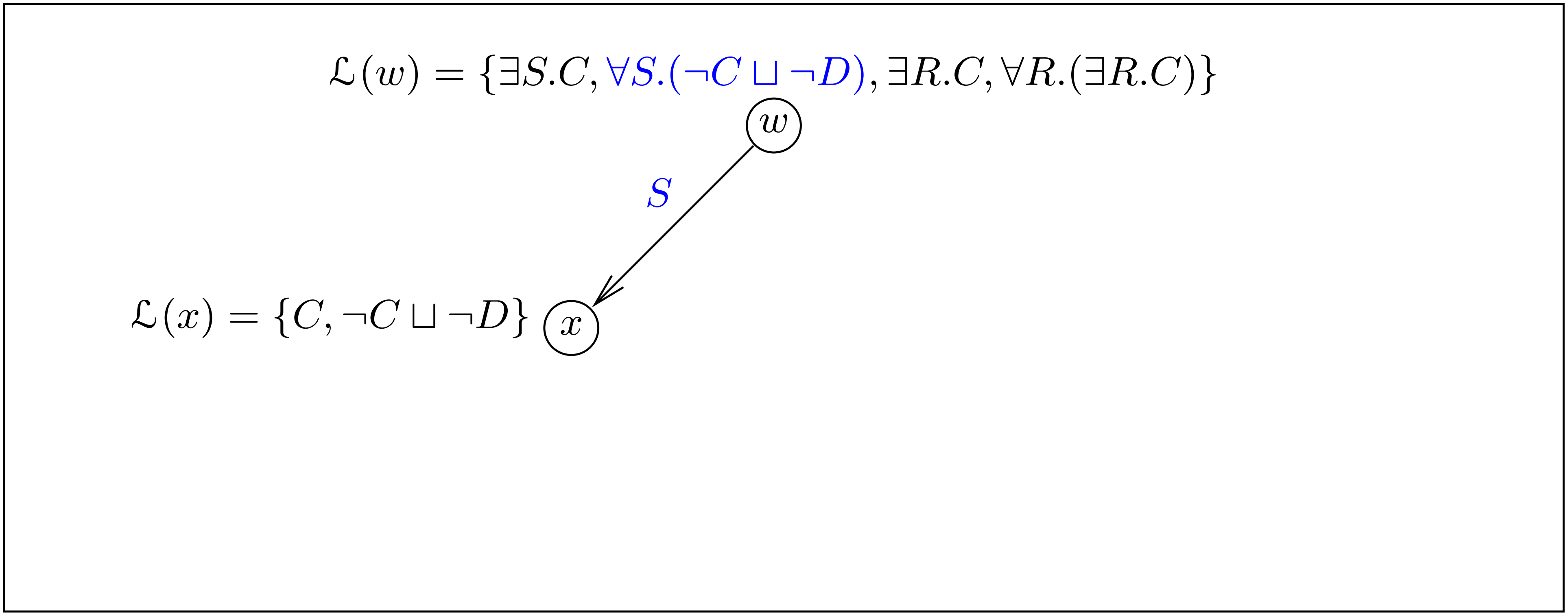
Tableaux Algorithm — Example

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



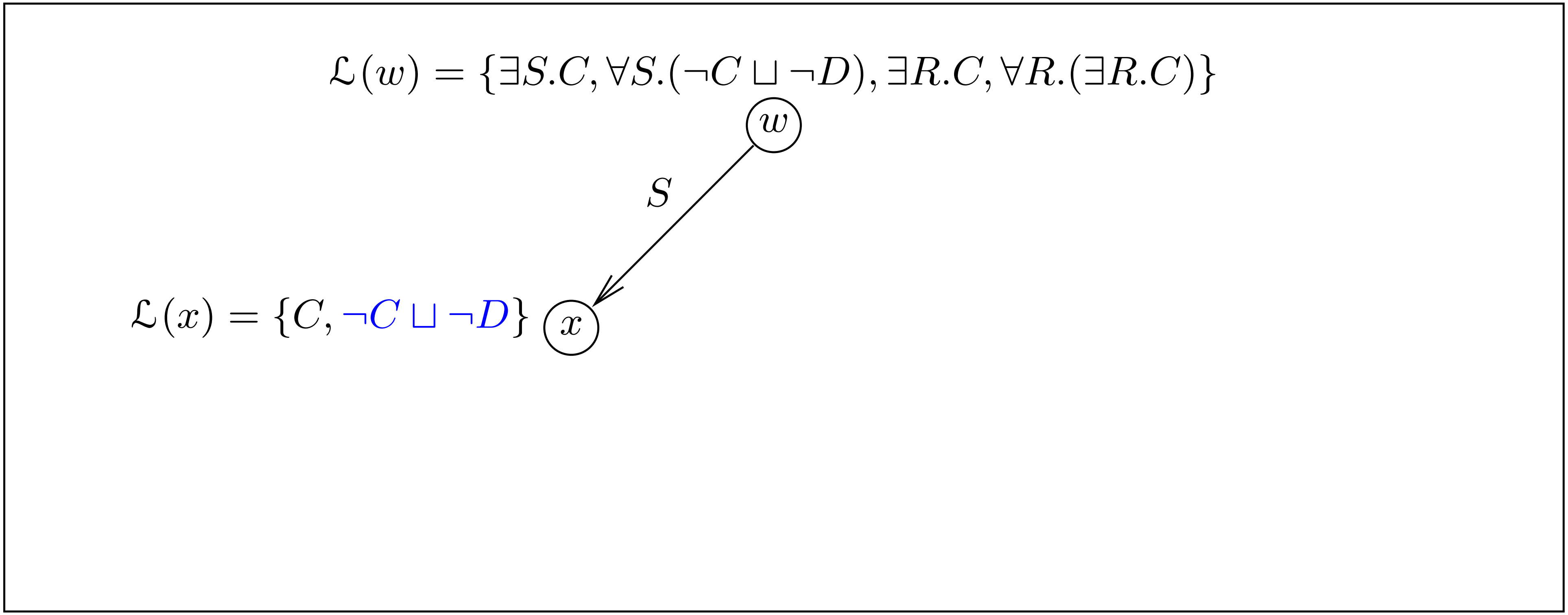
Tableaux Algorithm — Example

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



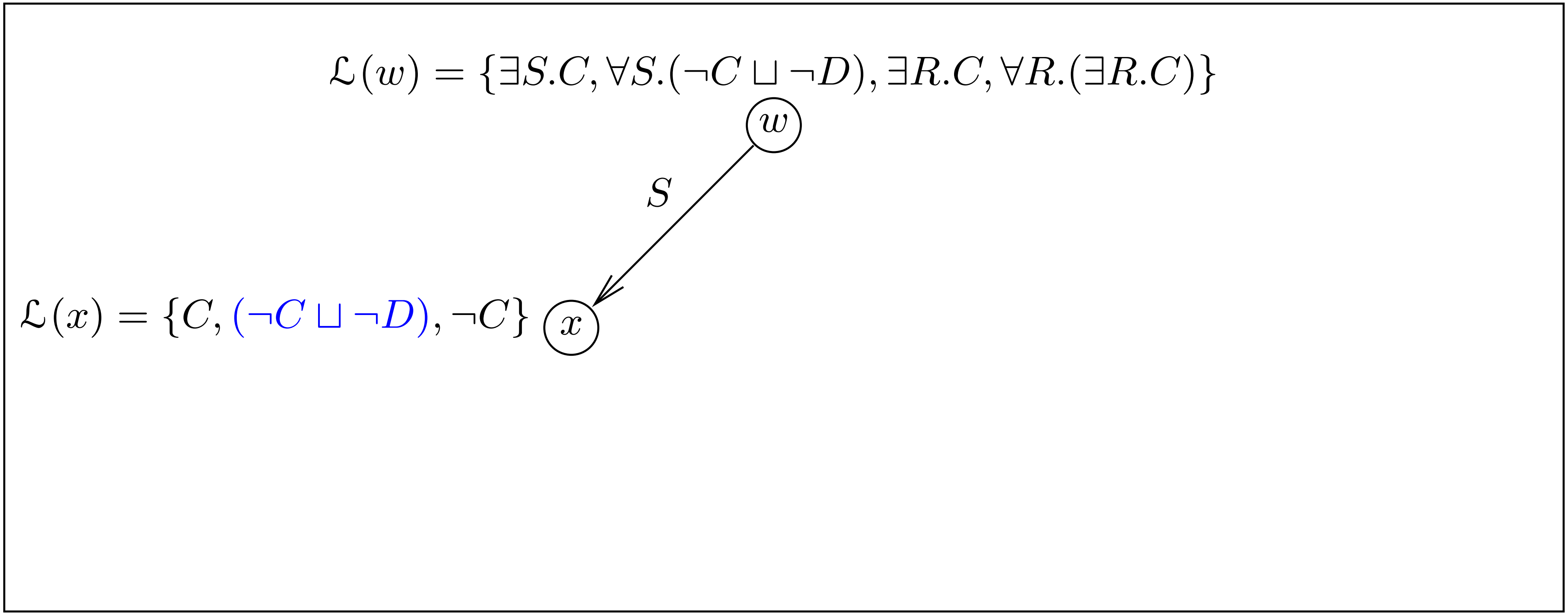
Tableaux Algorithm — Example

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 Algorithm — Example

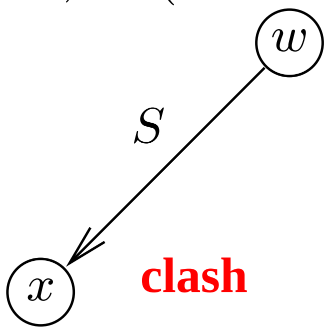
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

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

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

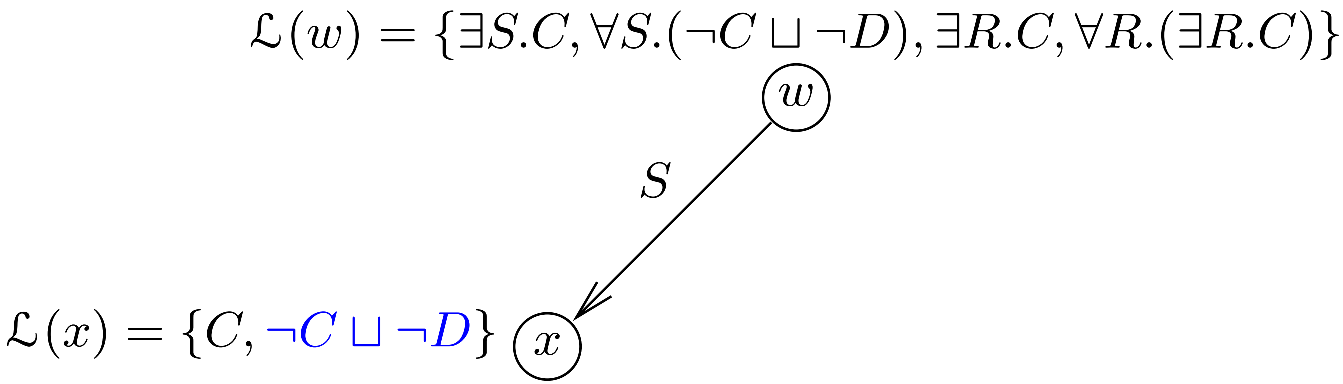


$$\mathcal{L}(x) = \{C, (\neg C \sqcup \neg D), \neg C\}$$

clash

Tableaux Algorithm — Example

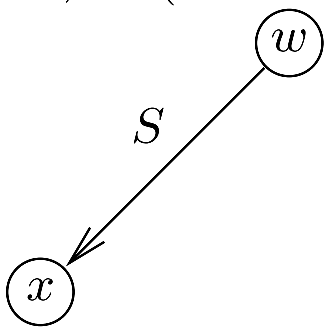
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Tableaux Algorithm — Example

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

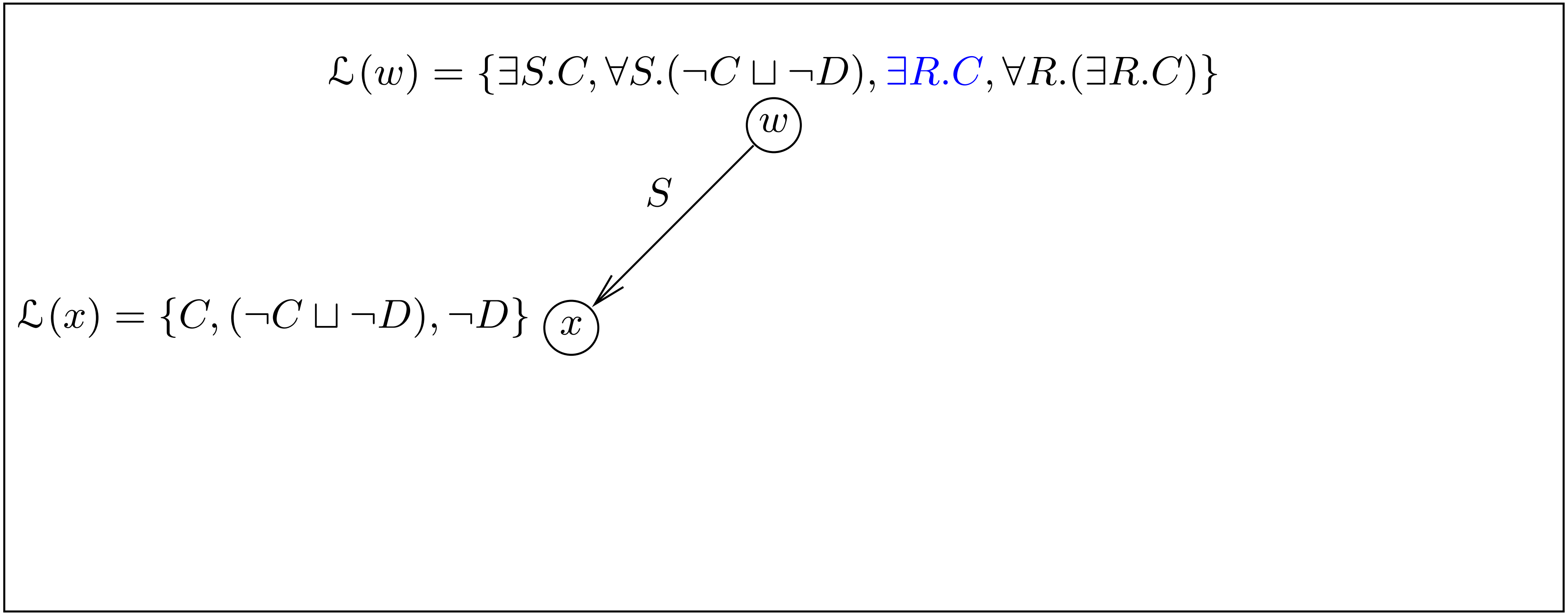
$$\mathcal{L}(w) = \{\exists S.C, \forall S.(\neg C \sqcup \neg D), \exists R.C, \forall R.(\exists R.C)\}$$



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

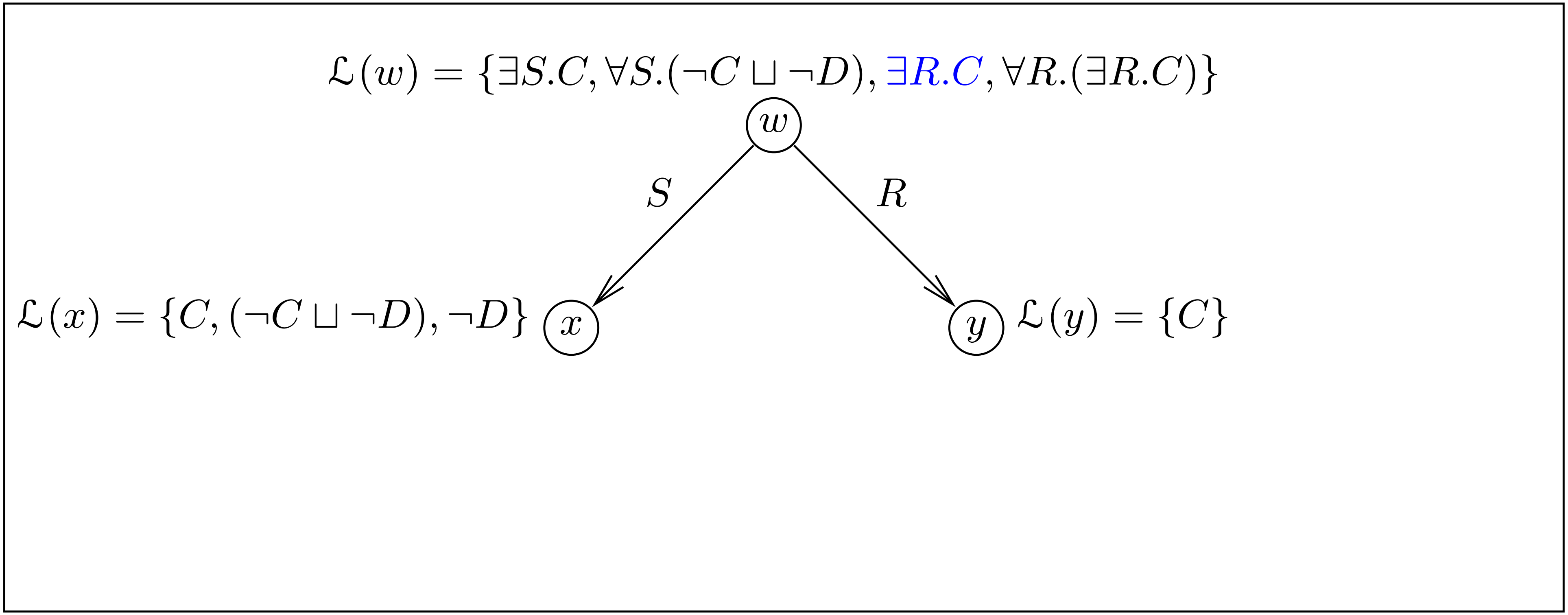
Tableaux Algorithm — Example

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



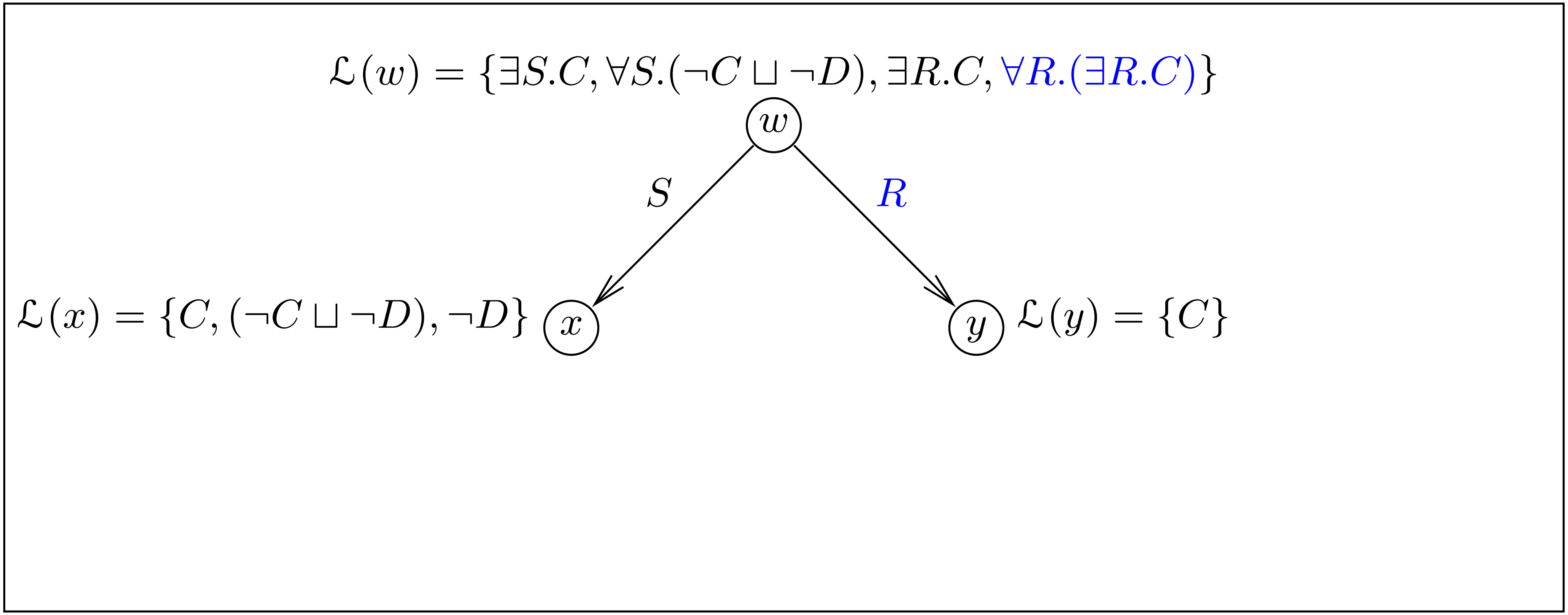
Tableaux Algorithm — Example

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



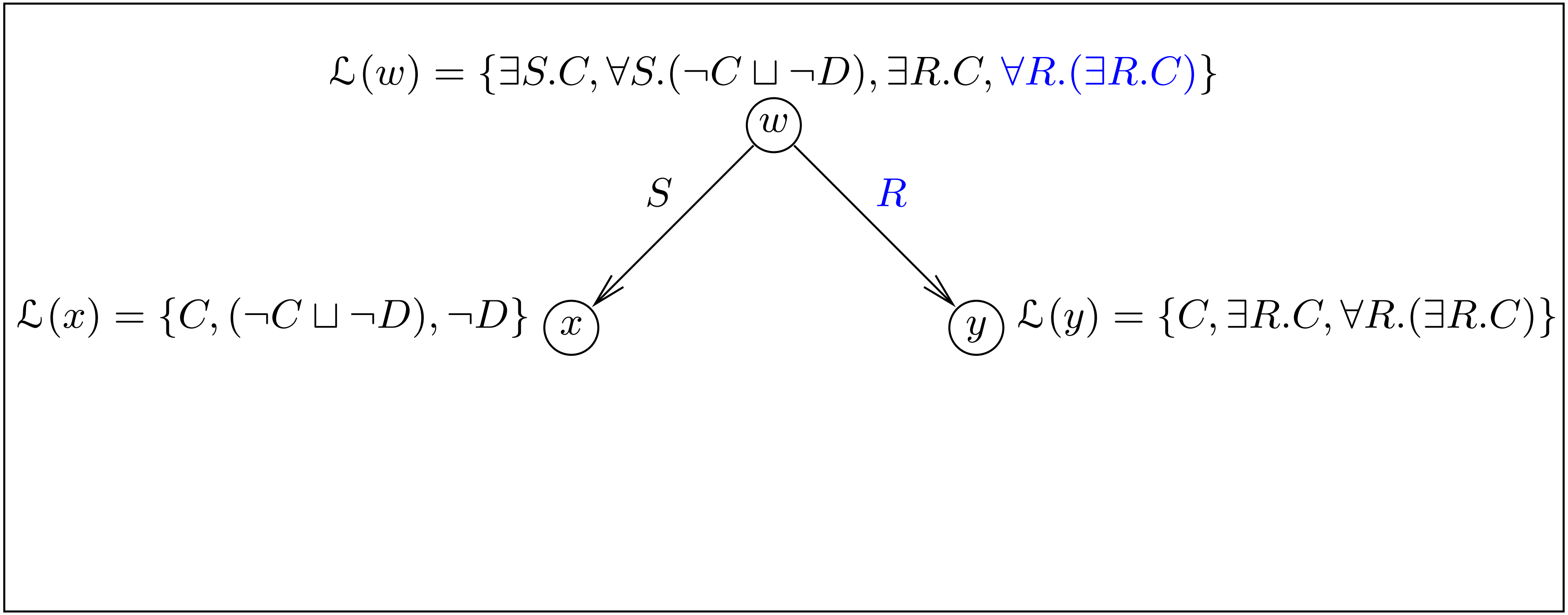
Tableaux Algorithm — Example

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



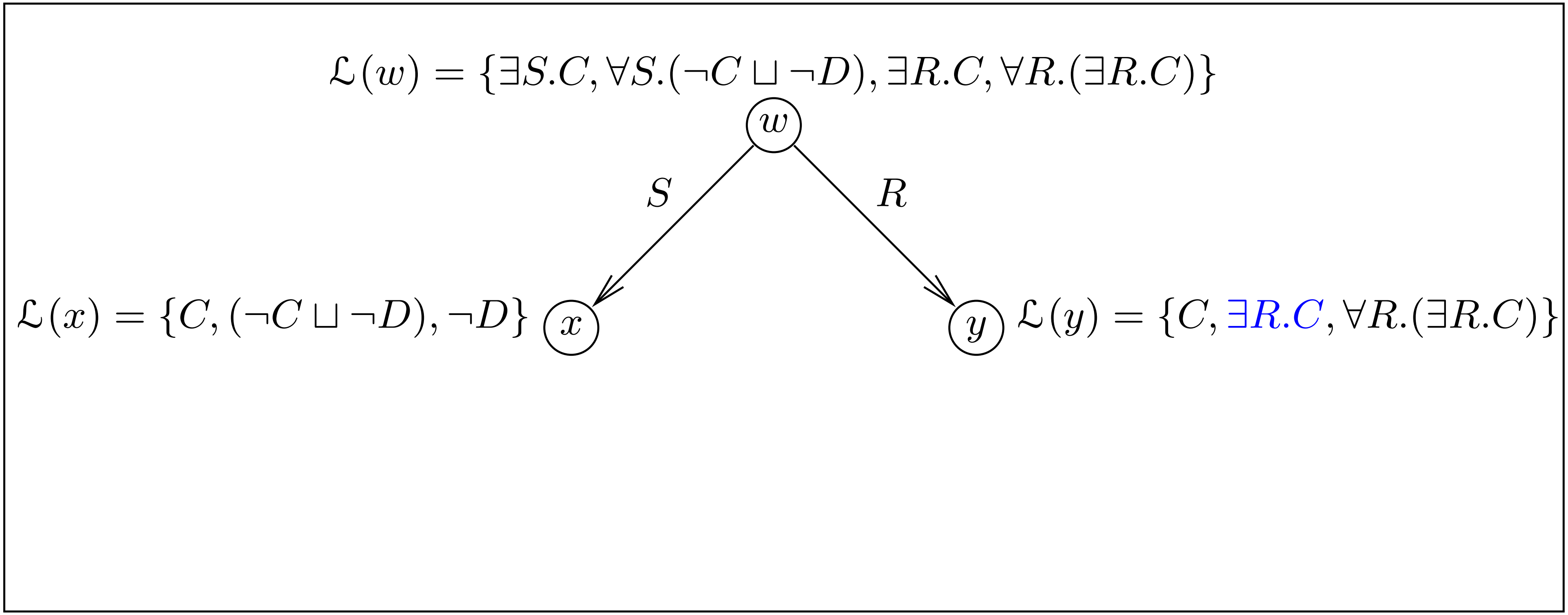
Tableaux Algorithm — Example

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



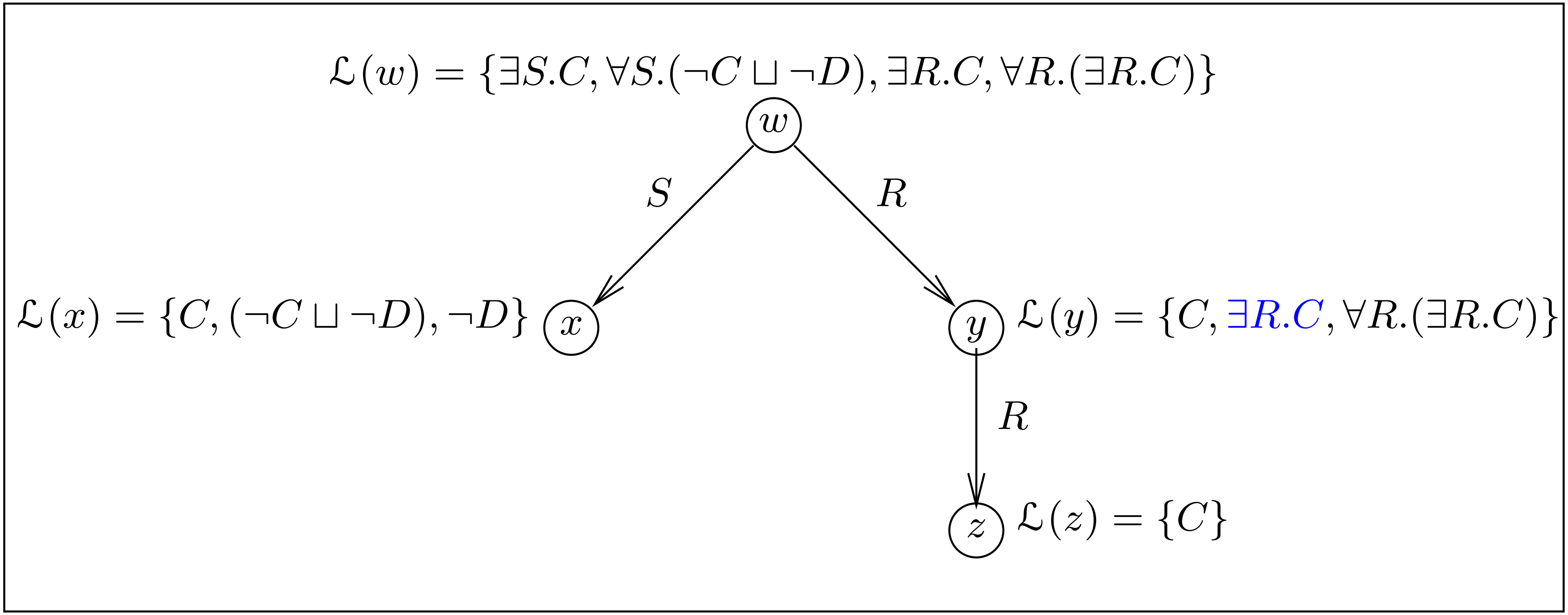
Tableaux Algorithm — Example

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



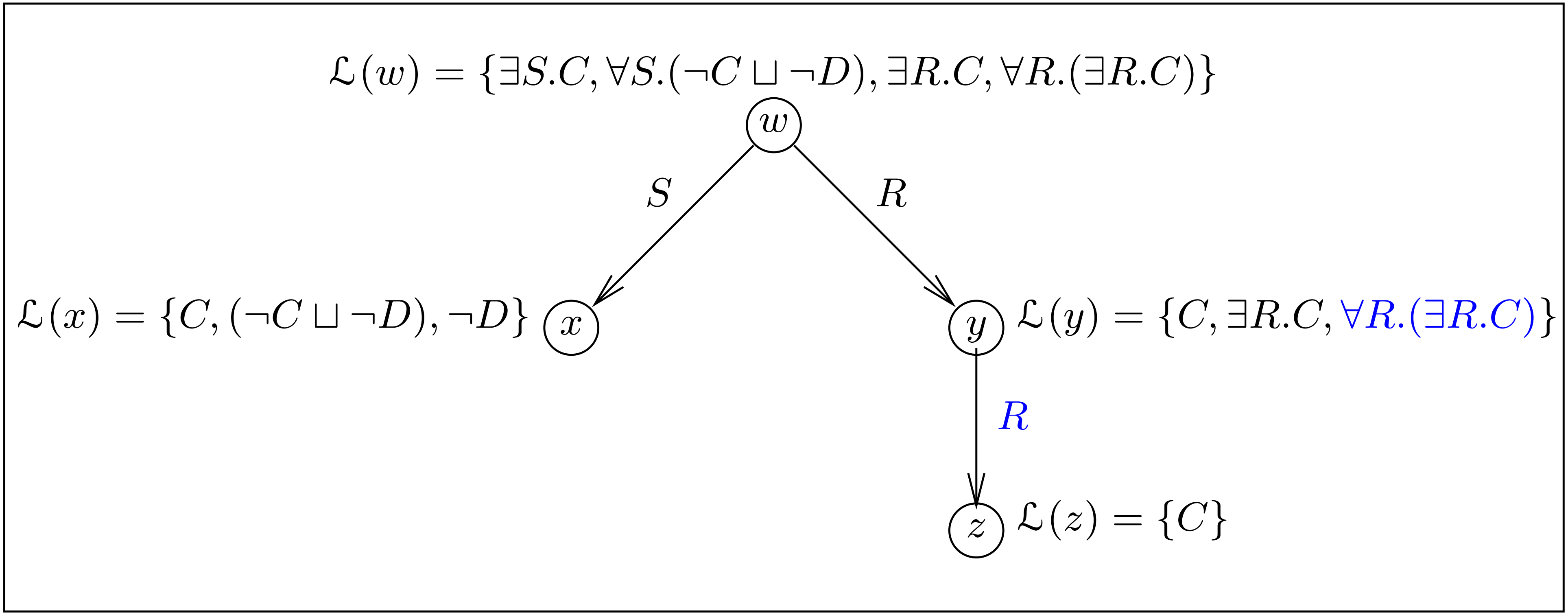
Tableaux Algorithm — Example

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



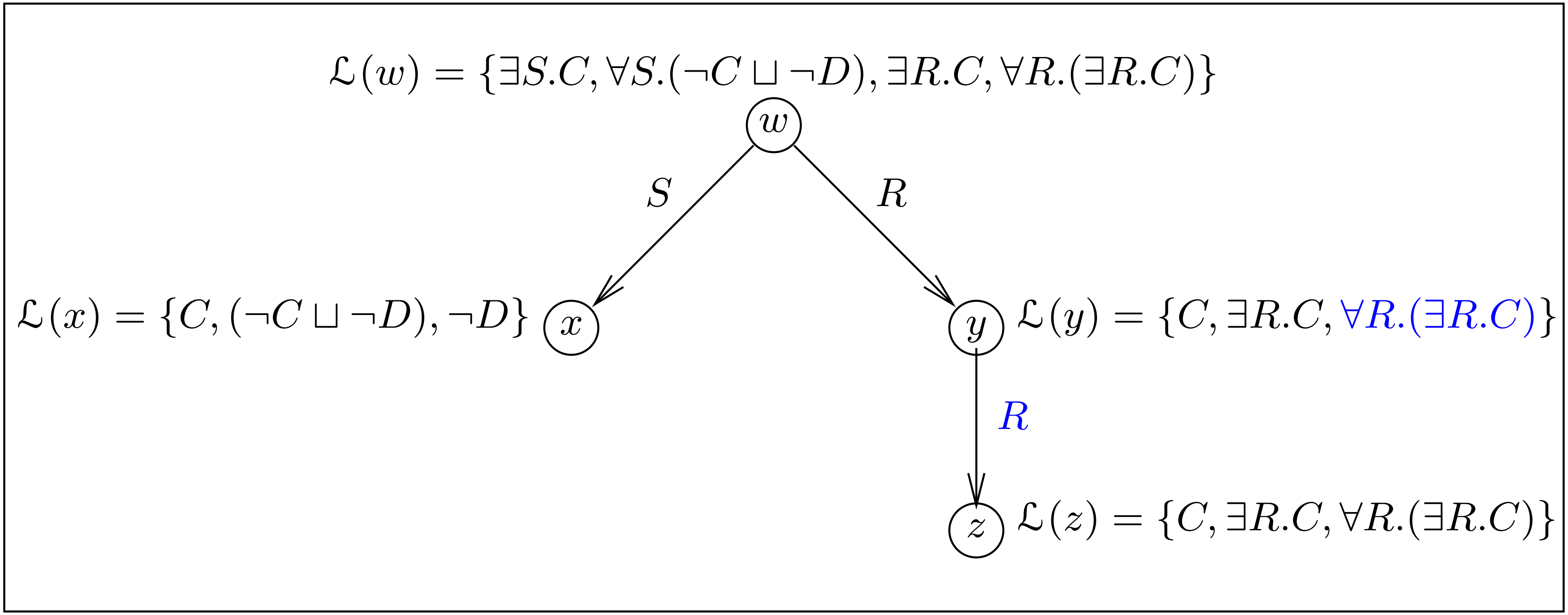
Tableaux Algorithm — Example

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



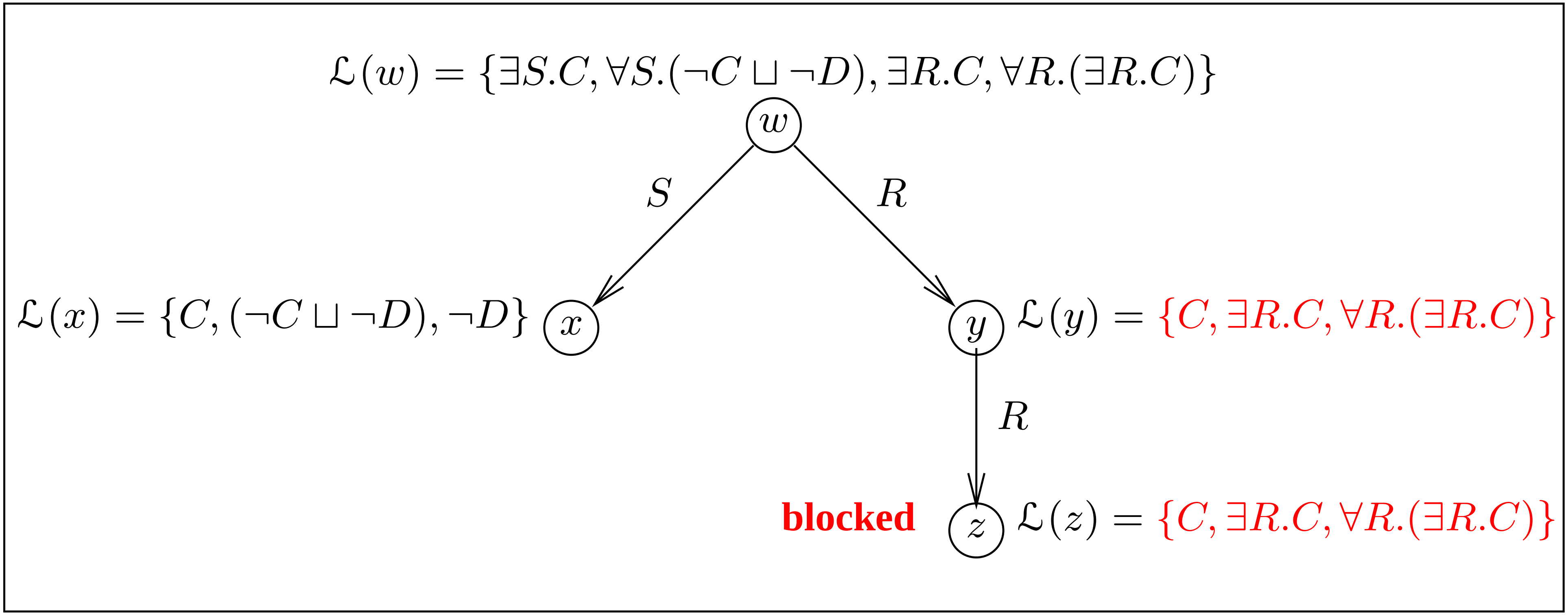
Tableaux Algorithm — Example

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



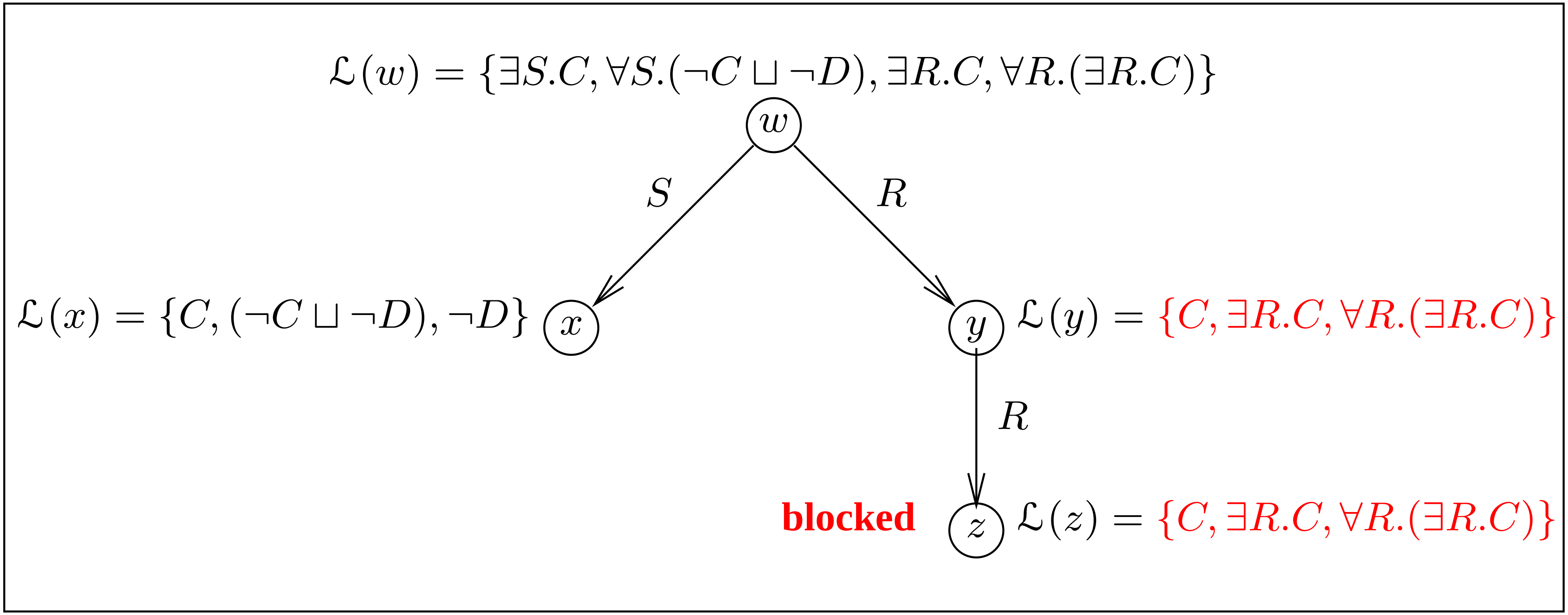
Tableaux Algorithm — Example

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 Algorithm — Example

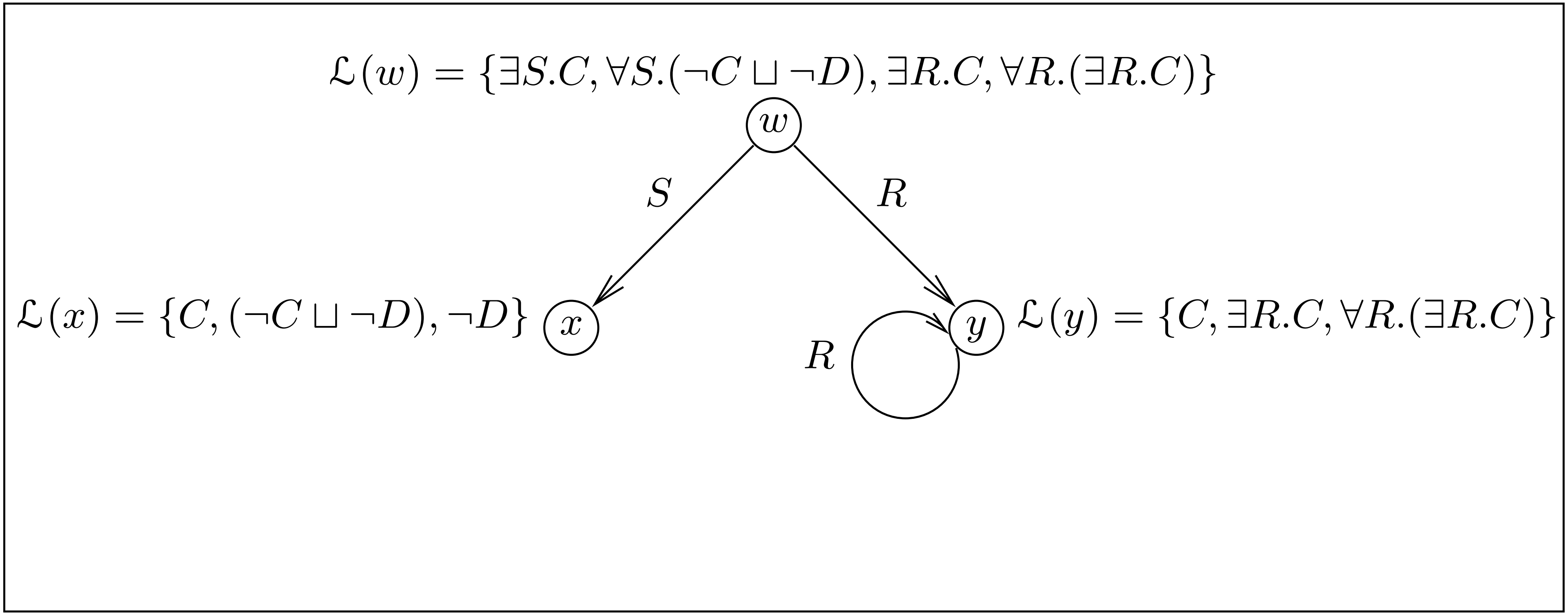
Test satisfiability of $\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R.(\exists R.C)\}$ where R is a **transitive** role



Concept is **satisfiable**: $\mathbf{T}$ corresponds to **model**

Tableaux Algorithm — Example

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**: $\mathbf{T}$ corresponds to **model**

More Advanced Techniques

More Advanced Techniques

Satisfiability w.r.t. a Terminology

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

More Advanced Techniques

Satisfiability w.r.t. a Terminology

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

More expressive DLs

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- ☞ For each axiom $C \sqsubseteq D \in \mathcal{T}$, 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 and datatypes
  - Aboxes
  - etc.

More Advanced Techniques

Satisfiability w.r.t. a Terminology

- ☞ For each axiom $C \sqsubseteq D \in \mathcal{T}$, 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 and datatypes
  - Aboxes
  - etc.
- ☞ Extend **expansion rules** and use more sophisticated **blocking** strategy

More Advanced Techniques

Satisfiability w.r.t. a Terminology

☞ For each axiom $C \sqsubseteq D \in \mathcal{T}$, 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 and datatypes
- Aboxes
- etc.

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

☞ **Forest** instead of Tree (for Aboxes)
- Root nodes correspond to individuals in Abox

Highly Optimised Implementation

Highly Optimised Implementation

- ☞ Naive implementation $\longrightarrow$ effective non-termination

Highly Optimised Implementation

- Naive implementation $\longrightarrow$ effective non-termination
- Modern systems include **MANY** optimisations

Highly Optimised Implementation

- ☞ Naive implementation $\longrightarrow$ effective non-termination
- ☞ Modern systems include **MANY** optimisations
- ☞ Optimised **classification** (compute partial ordering)
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order

Highly Optimised Implementation

- ☞ Naive implementation $\longrightarrow$ effective non-termination
- ☞ Modern systems include **MANY** optimisations
- ☞ Optimised **classification** (compute partial ordering)
  - Use enhanced traversal (exploit information from previous tests)
  - Use structural information to select classification order
- ☞ Optimised **subsumption** testing (search for models)
  - Normalisation and simplification of concepts
  - Absorption (simplification) of general axioms
  - Davis-Putnam style semantic branching search
  - Dependency directed backtracking
  - Caching of satisfiability results and (partial) models
  - Heuristic ordering of propositional and modal expansion
  - ...

Research Challenges

Challenges

Challenges

☞ **Increased expressive power**

- Existing DL systems implement (at most) $\mathcal{SHIQ}$
- DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals

Challenges

- **Increased expressive power**
  - Existing DL systems implement (at most) $\mathcal{SHIQ}$
  - DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals
- **Scalability**
  - Very large KBs
  - Reasoning with (very large numbers of) individuals

Challenges

- **Increased expressive power**
  - Existing DL systems implement (at most) $\mathcal{SHIQ}$
  - DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals
- **Scalability**
  - Very large KBs
  - Reasoning with (very large numbers of) individuals
- **Other reasoning tasks**
  - Querying
  - Matching
  - Least common subsumer
  - ...

Challenges

- **Increased expressive power**
  - Existing DL systems implement (at most) $\mathcal{SHIQ}$
  - DAML+OIL extends $\mathcal{SHIQ}$ with datatypes and nominals
- **Scalability**
  - Very large KBs
  - Reasoning with (very large numbers of) individuals
- **Other reasoning tasks**
  - Querying
  - Matching
  - Least common subsumer
  - . . .
- **Tools and Infrastructure**
  - Support for large scale ontological engineering and deployment

Increased Expressive Power: Datatypes

Increased Expressive Power: Datatypes

- **DAML+OIL** has simple form of datatypes
  - Unary predicates plus disjoint object-class/datatype domains

Increased Expressive Power: Datatypes

- **DAML+OIL** has simple form of datatypes
  - Unary predicates plus disjoint object-class/datatype domains
- Well understood **theoretically**
  - Existing work on **concrete domains** [Baader & Hanschke, Lutz]
  - Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]
  - Can use **hybrid reasoning** (DL reasoner + datatype "oracle")

Increased Expressive Power: Datatypes

- **DAML+OIL** has simple form of datatypes
  - Unary predicates plus disjoint object-class/datatype domains
- Well understood **theoretically**
  - Existing work on **concrete domains** [Baader & Hanschke, Lutz]
  - Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]
  - Can use **hybrid reasoning** (DL reasoner + datatype "oracle")
- May be **practically** challenging
  - All XMLS datatypes supported (?)

Increased Expressive Power: Datatypes

- **DAML+OIL** has simple form of datatypes
  - Unary predicates plus disjoint object-class/datatype domains
- Well understood **theoretically**
  - Existing work on **concrete domains** [Baader & Hanschke, Lutz]
  - Algorithm already known for $\mathcal{SHOQ}(\mathbf{D})$ [Horrocks & Sattler]
  - Can use **hybrid reasoning** (DL reasoner + datatype "oracle")
- May be **practically** challenging
  - All XMLS datatypes supported (?)
- Already seeing some (partial) **implementations**
  - Cerebra system (Network Inference), Racer system (Hamburg)

Increased Expressive Power: Nominals

Increased Expressive Power: Nominals

- DAML+OIL **oneOf** constructor equivalent to hybrid logic **nominals**
  - Extensionally defined concepts, e.g., EU $\equiv \{\text{France}, \text{Italy}, \ldots\}$

Increased Expressive Power: Nominals

- DAML+OIL **oneOf** constructor equivalent to hybrid logic **nominals**
  - Extensionally defined concepts, e.g., EU $\equiv \{\text{France}, \text{Italy}, \ldots\}$
- 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

- ☞ DAML+OIL **oneOf** constructor equivalent to hybrid logic **nominals**
  - Extensionally defined concepts, e.g., EU $\equiv \{\text{France}, \text{Italy}, \ldots\}$
- ☞ 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: $\{Spy\} \sqsubseteq \leqslant n R^-$
  - ?? **automata** based algorithms ??

Increased Expressive Power: Nominals

- DAML+OIL **oneOf** constructor equivalent to hybrid logic **nominals**
  - Extensionally defined concepts, e.g., EU $\equiv \{\text{France}, \text{Italy}, \ldots\}$
- 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: $\{Spy\} \sqsubseteq \leqslant n R^{-}$
  - ?? **automata** based algorithms ??
- **Standard solution** is weaker semantics for nominals
  - Treat nominals as (disjoint) primitive classes
  - Loose some inferential power, e.g., w.r.t. max cardinality

Scalability

Scalability

- ☞ Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)

Scalability

- Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)
- Web ontologies may grow **very large**

Scalability

- Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)
- Web ontologies may grow **very large**
- Good **empirical evidence** of scalability/tractability for DL systems
  - E.g., 5,000 (complex) classes; 100,000+ (simple) classes

Scalability

- Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)
- Web ontologies may grow **very large**
- Good **empirical evidence** of scalability/tractability for DL systems
  - E.g., 5,000 (complex) classes; 100,000+ (simple) classes
- But evidence mostly w.r.t. $\mathcal{SHF}$ (no inverse)

Scalability

- Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)
- Web ontologies may grow **very large**
- Good **empirical evidence** of scalability/tractability for DL systems
  - E.g., 5,000 (complex) classes; 100,000+ (simple) classes
- But evidence mostly w.r.t. $\mathcal{SHF}$ (no inverse)
- **Problems** can arise when $\mathcal{SHF}$ extended to $\mathcal{SHIQ}$
  - Important **optimisations** no longer (fully) work

Scalability

- Reasoning **hard** (ExpTime) even without nominals (i.e., $\mathcal{SHIQ}$)
- Web ontologies may grow **very large**
- Good **empirical evidence** of scalability/tractability for DL systems
  - E.g., 5,000 (complex) classes; 100,000+ (simple) classes
- But evidence mostly w.r.t. $\mathcal{SHF}$ (no inverse)
- **Problems** can arise when $\mathcal{SHF}$ extended to $\mathcal{SHIQ}$
  - Important **optimisations** no longer (fully) work
- Reasoning with **individuals**
  - **Deployment** of web ontologies will mean reasoning with (possibly very large numbers of) individuals/tuples
  - Unlikely that standard **Abox** techniques will be able to cope

Other Reasoning Tasks

Other Reasoning Tasks

☞ **Querying**

- Retrieval and instantiation wont be sufficient
- Minimum requirement will be **DB style query language**
- May also need “what can I say about x?” style of query

Other Reasoning Tasks

- **Querying**
  - Retrieval and instantiation wont be sufficient
  - Minimum requirement will be **DB style query language**
  - May also need “what can I say about x?” style of query
- **Explanation**
  - To support ontology design
  - Justifications and proofs (e.g., of query results)

Other Reasoning Tasks

- **Querying**
  - Retrieval and instantiation wont be sufficient
  - Minimum requirement will be **DB style query language**
  - May also need “what can I say about x?” style of query
- **Explanation**
  - To support ontology design
  - Justifications and proofs (e.g., of query results)
- “**Non-Standard Inferences**”, e.g., LCS, matching
  - To support ontology integration
  - To support “bottom up” design of ontologies

Summary

Summary

☞ **Description Logics** are family of logical KR formalisms

Summary

☞ **Description Logics** are family of logical KR formalisms

☞ **Applications** of DLs include DataBases and **Semantic Web**

- Ontologies will provide vocabulary for semantic markup
- DAML+OIL web ontology language based on $\mathcal{SHIQ}$ DL
- Set to become W3C standard (OWL) & already widely adopted
- Use of DL provides formal foundations and reasoning support

Summary

- ☞ **Description Logics** are family of logical KR formalisms
- ☞ **Applications** of DLs include DataBases and **Semantic Web**
  - Ontologies will provide vocabulary for semantic markup
  - DAML+OIL web ontology language based on $\mathcal{SHIQ}$ DL
  - Set to become W3C standard (OWL) & already widely adopted
  - Use of DL provides formal foundations and reasoning support
- ☞ **DL Reasoning** based on tableau algorithms

Summary

- **Description Logics** are family of logical KR formalisms
- **Applications** of DLs include DataBases and **Semantic Web**
  - Ontologies will provide vocabulary for semantic markup
  - DAML+OIL web ontology language based on $\mathcal{SHIQ}$ DL
  - Set to become W3C standard (OWL) & already widely adopted
  - Use of DL provides formal foundations and reasoning support
- **DL Reasoning** based on tableau algorithms
- **Highly Optimised** implementations used in DL systems

Summary

- **Description Logics** are family of logical KR formalisms
- **Applications** of DLs include DataBases and **Semantic Web**
  - Ontologies will provide vocabulary for semantic markup
  - DAML+OIL web ontology language based on $\mathcal{SHIQ}$ DL
  - Set to become W3C standard (OWL) & already widely adopted
  - Use of DL provides formal foundations and reasoning support
- **DL Reasoning** based on tableau algorithms
- **Highly Optimised** implementations used in DL systems
- **Challenges** remain
  - Reasoning with full DAML+OIL/OWL language
  - (Convincing) demonstration(s) of scalability
  - New reasoning tasks
  - Development of (high quality) tools and infrastructure

Acknowledgements

Acknowledgements

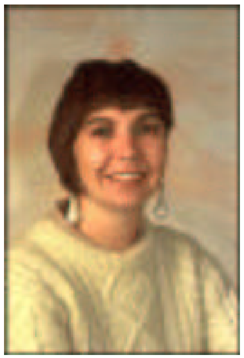
☞ Members of the OIL and DAML+OIL development teams, in particular Dieter Fensel and Frank van Harmelen (Amsterdam) and Peter Patel-Schneider (Bell Labs)

Acknowledgements

- Members of the OIL and DAML+OIL development teams, in particular Dieter Fensel and Frank van Harmelen (Amsterdam) and Peter Patel-Schneider (Bell Labs)
- Franz Baader, Uli Sattler and Stefan Tobies (Dresden)

Acknowledgements

- Members of the OIL and DAML+OIL development teams, in particular Dieter Fensel and Frank van Harmelen (Amsterdam) and Peter Patel-Schneider (Bell Labs)
- Franz Baader, Uli Sattler and Stefan Tobies (Dresden)
- Members of the Information Management, Medical Informatics and Formal Methods Groups at the University of Manchester



Resources

Slides from this talk

http://www.cs.man.ac.uk/~horrocks/Slides/ed02.pdf

FaCT system (open source)

http://www.cs.man.ac.uk/FaCT/

OilEd (open source)

http://oiled.man.ac.uk/

OIL

http://www.ontoknowledge.org/oil/

DAML+OIL

http://www.w3c.org/Submission/2001/12/

W3C Web-Ontology (WebOnt) working group (OWL)

http://www.w3.org/2001/sw/WebOnt/ **DL Handbook** — available autumn 2002 from Cambridge University Press

Select Bibliography

I. Horrocks. DAML+OIL: a reason-able web ontology language. In *Proc. of EDBT 2002*, number 2287 in Lecture Notes in Computer Science, pages 2–13. Springer-Verlag, Mar. 2002.

I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In *Proc. of AAAI 2002*, 2002. To appear.

I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In I. Horrocks and J. Hendler, editors, *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

Select Bibliography

I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ description logic. In B. Nebel, editor, *Proc. of IJCAI-01*, pages 199–204. Morgan Kaufmann, 2001.

F. Baader, S. Brandt, and R. Küsters. Matching under side conditions in description logics. In B. Nebel, editor, *Proc. of IJCAI-01*, pages 213–218, Seattle, Washington, 2001. Morgan Kaufmann.

A. Borgida, E. Franconi, and I. Horrocks. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of ECAI 2000*, pages 209–213. IOS Press, 2000.

D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DWDM'99)*, 1999.