On Axiomatic Rejection for the Description Logic $\mathcal{ALC}$

Hans Tompits

Vienna University of Technology
Institute of Information Systems
Knowledge-Based Systems Group

Joint work with Gerald Berger

Context

- The traditional view about proof calculi is that they are *assertional*—their aim is to axiomatise the *valid propositions* of a logic.

Context

- The traditional view about proof calculi is that they are *assertional*—their aim is to axiomatise the *valid propositions* of a logic.
- But we can also have a complementary view:
  - Instead of axiomatising the valid sentences we may axiomatise the *invalid* ones.

Context

- The traditional view about proof calculi is that they are *assertional*—their aim is to axiomatise the *valid propositions* of a logic.
- But we can also have a complementary view:
  - Instead of axiomatising the valid sentences we may axiomatise the *invalid* ones.
  - In such a system, false propositions are deduced from other (elementary) false ones.

Context

- The traditional view about proof calculi is that they are *assertional*—their aim is to axiomatise the *valid propositions* of a logic.
- But we can also have a complementary view:
  - Instead of axiomatising the valid sentences we may axiomatise the *invalid* ones.
  - In such a system, false propositions are deduced from other (elementary) false ones.

➥ Calculi axiomatising the invalid sentences of a logic are called *rejection systems* or *complementary calculi*.

Context (ctd.)

“Ich bin der Geist der stets verneint!
Und das mit Recht; denn alles, was entsteht,
Ist wert, dass es zugrunde geht.”

(“I am the spirit, ever, that denies!
And rightly so; since everything created,
In turn deserves to be annihilated.”)

–J.W. von Goethe, Faust I

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*
    - provide the formal underpinning for semantic-web reasoning.

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*
    - provide the formal underpinning for semantic-web reasoning.
- Our calculus axiomatises *concept non-subsumption*.
  - That is, a sequent $C \dashv D$ is provable in our calculus iff $C \sqsubseteq D$ does not hold.

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*
    - provide the formal underpinning for semantic-web reasoning.
- Our calculus axiomatises *concept non-subsumption*.
  - That is, a sequent $C \dashv D$ is provable in our calculus iff $C \sqsubseteq D$ does not hold.
- We also analyse the relationship between our calculus and a well-known tableau procedure for $\mathcal{ALC}$.

Main Contributions

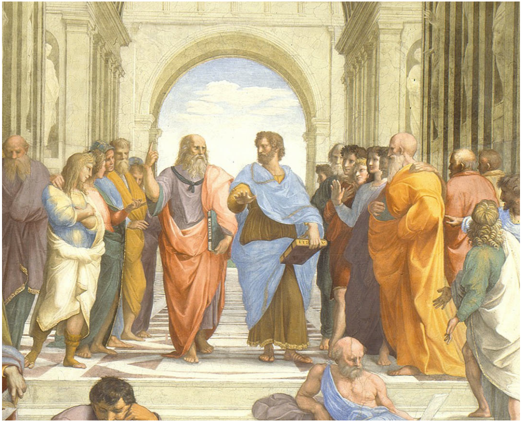
- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*
    - provide the formal underpinning for semantic-web reasoning.
- Our calculus axiomatises *concept non-subsumption*.
  - That is, a sequent $C \dashv D$ is provable in our calculus iff $C \sqsubseteq D$ does not hold.
- We also analyse the relationship between our calculus and a well-known tableau procedure for $\mathcal{ALC}$.
- Finally, we also obtain a calculus for the *multi-modal version of modal logic* $\mathbf{K}$ by the relation of $\mathcal{ALC}$ with this logic

Main Contributions

- We introduce a *Gentzen-type rejection systems* for description logic $\mathcal{ALC}$.
  - Gentzen-type systems are well-known calculi optimised for *proof search*.
  - Description logics are important knowledge-representation languages for modelling *ontologies*
    - provide the formal underpinning for semantic-web reasoning.
- Our calculus axiomatises *concept non-subsumption*.
  - That is, a sequent $C \dashv D$ is provable in our calculus iff $C \sqsubseteq D$ does not hold.
- We also analyse the relationship between our calculus and a well-known tableau procedure for $\mathcal{ALC}$.
- Finally, we also obtain a calculus for the *multi-modal version of modal logic* $\mathbf{K}$ by the relation of $\mathcal{ALC}$ with this logic
  - ➥ generalises a rejection calculus for standard $\mathbf{K}$ by Goranko (1994).

Historical Remarks

Investigation of invalid arguments traces back to *Aristotle* in his analysis of *logical fallacies* in *On Sophistical Refutations* of the *Organon*.

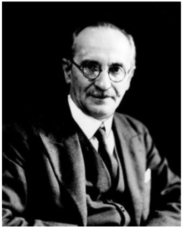


Raphael's Scuola di Atene

Historical Remarks (ctd.)

- The first system for axiomatic rejection was introduced by *Jan Łukasiewicz* in 1957 in his book

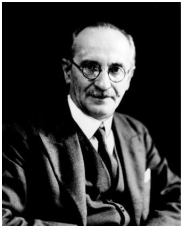
  "Aristotle's syllogistic from the standpoint of modern formal logic"



Historical Remarks (ctd.)

- The first system for axiomatic rejection was introduced by *Jan Łukasiewicz* in 1957 in his book

  "Aristotle's syllogistic from the standpoint of modern formal logic"



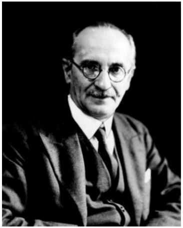
  - There, he axiomatised invalid syllogisms of Aristotle by means of a Hilbert-type system using the detachment rule

    if $\varphi \supset \psi$ is asserted and ψ is rejected, then φ is rejected too.

Historical Remarks (ctd.)

- The first system for axiomatic rejection was introduced by *Jan Łukasiewicz* in 1957 in his book

  "Aristotle's syllogistic from the standpoint of modern formal logic"



  - There, he axiomatised invalid syllogisms of Aristotle by means of a Hilbert-type system using the detachment rule

    if $\varphi \supset \psi$ is asserted and ψ is rejected, then φ is rejected too.

- Subsequently, other rejection systems were introduced for intuitionistic logic, different modal logics, and many-valued logics.

Relevance of Axiomatic Rejection

- Complementary calculi are relevant for axiomatising *nonmonotonic logics*.

Relevance of Axiomatic Rejection

- Complementary calculi are relevant for axiomatising *nonmonotonic logics*.
  - E.g., *default rule* "if A and no evidence for B then C" amounts to inference

$$\frac{\vdash A \qquad \dashv B}{\vdash C}$$

Relevance of Axiomatic Rejection

- Complementary calculi are relevant for axiomatising *nonmonotonic logics*.
  - E.g., *default rule* "if A and no evidence for B then C" amounts to inference

$$\frac{\vdash A \qquad \dashv B}{\vdash C}$$

  - Indeed, Gentzen-type axiomatisations of central nonmonotonic logics (like default logic, circumscription, autoepistemic logic) rely on rejection calculi (Bonatti & Olivetti, 2002).

Relevance of Axiomatic Rejection

➤ Complementary calculi are relevant for axiomatising *nonmonotonic logics*.

- E.g., *default rule* "if A and no evidence for B then C" amounts to inference

$$\frac{\vdash A \qquad \dashv B}{\vdash C}$$

- Indeed, Gentzen-type axiomatisations of central nonmonotonic logics (like default logic, circumscription, autoepistemic logic) rely on rejection calculi (Bonatti & Olivetti, 2002).

➥ Rejection calculi become relevant in studying proof systems of *nonmonotonic extensions of description logics*.

Relevance of Axiomatic Rejection

➤ Complementary calculi are relevant for axiomatising *nonmonotonic logics*.

- E.g., *default rule* "if A and no evidence for B then C" amounts to inference

$$\frac{\vdash A \qquad \dashv B}{\vdash C}$$

- Indeed, Gentzen-type axiomatisations of central nonmonotonic logics (like default logic, circumscription, autoepistemic logic) rely on rejection calculi (Bonatti & Olivetti, 2002).

➥ Rejection calculi become relevant in studying proof systems of *nonmonotonic extensions of description logics*.

☞ Nonmonotonic DLs are the topic of recent investigations, e.g., by Casini et al. (DL 2013) and Giordano et al. (AIJ, 2013).

Description Logic $\mathcal{ALC}$—Syntax

➤ The *vocabulary* of $\mathcal{ALC}$ includes the following elements:

- concept names $A, B, \ldots,$
- role names $p, q, r, \ldots,$
- individual names $a, b, c, \ldots$,

Description Logic $\mathcal{ALC}$—Syntax

➤ The *vocabulary* of $\mathcal{ALC}$ includes the following elements:

- concept names $A, B, \ldots,$
- role names $p, q, r, \ldots,$
- individual names $a, b, c, \ldots$,
- concept intersection $\sqcap$,
- concept union $\sqcup$,
- concept negation $\neg$,
- value restriction $\forall$,
- existential restriction $\exists$.

Description Logic $\mathcal{ALC}$—Syntax

- The *vocabulary* of $\mathcal{ALC}$ includes the following elements:
  - concept names $A, B, \ldots,$
  - role names $p, q, r, \ldots,$
  - individual names $a, b, c, \ldots$,
  - concept intersection $\sqcap$,
  - concept union $\sqcup$,
  - concept negation $\neg$,
  - value restriction $\forall$,
  - existential restriction $\exists$.
- Syntax of $\mathcal{ALC}$-concepts:

$$C ::= A \mid C \sqcap C \mid C \sqcup C \mid \neg C \mid \exists r.C \mid \forall r.C \mid \bot \mid \top$$

  - A denotes a concept name, while r denotes a role name,

Description Logic $\mathcal{ALC}$—Semantics

- An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where
  - $\Delta^{\mathcal{I}}$ is a non-empty set, called *domain*,
  - $\cdot^{\mathcal{I}}$ is a mapping ensuring that
    - every concept name A is mapped to some subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
    - every role name r is mapped to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Description Logic $\mathcal{ALC}$—Semantics

- An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where
  - $\Delta^{\mathcal{I}}$ is a non-empty set, called *domain*,
  - $\cdot^{\mathcal{I}}$ is a mapping ensuring that
    - every concept name A is mapped to some subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
    - every role name r is mapped to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.
  - $\cdot^{\mathcal{I}}$ satisfies the usual truth conditions concerning the concept constructors;

Description Logic $\mathcal{ALC}$—Semantics

- An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where
  - $\Delta^{\mathcal{I}}$ is a non-empty set, called *domain*,
  - $\cdot^{\mathcal{I}}$ is a mapping ensuring that
    - every concept name A is mapped to some subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
    - every role name r is mapped to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.
  - $\cdot^{\mathcal{I}}$ satisfies the usual truth conditions concerning the concept constructors; the semantics of the quantifiers is given by
    - $(\forall r.C)^{\mathcal{I}} = \{x \mid \forall y : (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$,
    - $(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$.

    That is:

    $(\forall r.C)$ corresponds to $\forall y (R(x, y) \supset C(y))$;
    $(\exists r.C)$ corresponds to $\exists y (R(x, y) \wedge C(y))$.

Description Logic $\mathcal{ALC}$—Semantics

- An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where
  - $\Delta^{\mathcal{I}}$ is a non-empty set, called *domain*,
  - $\cdot^{\mathcal{I}}$ is a mapping ensuring that
    - every concept name A is mapped to some subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
    - every role name r is mapped to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.
  - $\cdot^{\mathcal{I}}$ satisfies the usual truth conditions concerning the concept constructors; the semantics of the quantifiers is given by
    - $(\forall r.C)^{\mathcal{I}} = \{x \mid \forall y : (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$,
    - $(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$.

    That is:

    $(\forall r.C)$ corresponds to $\forall y(R(x, y) \supset C(y))$;
    $(\exists r.C)$ corresponds to $\exists y(R(x, y) \wedge C(y))$.
- A concept C is satisfiable iff there exists a finite *tree-shaped* interpretation $\mathcal{T}$ such that $v_0 \in C^{\mathcal{T}}$, where v_0 is the root of $\mathcal{T}$.

Description Logic $\mathcal{ALC}$—Semantics (ctd.)

- A *general concept inclusion* (*GCI*) is an expression of the form $C \sqsubseteq D$, where C and D are concepts.

Description Logic $\mathcal{ALC}$—Semantics (ctd.)

- A *general concept inclusion* (*GCI*) is an expression of the form $C \sqsubseteq D$, where C and D are concepts.
- An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Description Logic $\mathcal{ALC}$—Semantics (ctd.)

- A *general concept inclusion* (*GCI*) is an expression of the form $C \sqsubseteq D$, where C and D are concepts.
- An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- A concept D *subsumes* a concept C if every interpretation satisfies the GCI $C \sqsubseteq D$.

Description Logic $\mathcal{ALC}$—Semantics (ctd.)

- A *general concept inclusion (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts.
- An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- A concept D *subsumes* a concept C if every interpretation satisfies the GCI $C \sqsubseteq D$.

➥ In what follows, we introduce the sequential rejection system $\mathbf{SC}^{c}_{\mathcal{ALC}}$ which axiomatises *non-subsumption*.

Rejection Calculus $\mathbf{SC}^c_{\mathcal{ALC}}$

- An *anti-sequent* is a pair $\Gamma \dashv \Delta$, where Γ and Δ are finite multi-sets of concepts.

Rejection Calculus $\mathbf{SC}^{c}_{\mathcal{ALC}}$

- An *anti-sequent* is a pair $\Gamma \dashv \Delta$, where Γ and Δ are finite multi-sets of concepts.
- An interpretation *refutes* an anti-sequent $\Gamma \dashv \Delta$ if it does not satisfy the GCI

$$\bigsqcap_{\gamma \in \Gamma} \gamma \sqsubseteq \bigsqcup_{\delta \in \Delta} \delta;$$

(the empty concept intersection is $\top$; the empty concept union is $\bot$).

Rejection Calculus $\mathbf{SC}^c_{\mathcal{ALC}}$ (ctd.)

- Axioms of $\mathbf{SC}^c_{\mathcal{ALC}}$:
  - any anti-sequent

$$\Gamma_0 \dashv \Delta_0$$

    s.t. $\Gamma_0 \cap \Delta_0 = \emptyset$, where Γ_0 and Δ_0 consist of concept names only;
  - any anti-sequent of form

$$\forall r_1.\Gamma_1, \ldots, \forall r_n.\Gamma_n \dashv \exists r_1.\Delta_1, \ldots, \exists r_n.\Delta_n.$$

Rejection Calculus $\mathbf{SC}^c_{\mathcal{ALC}}$ (ctd.)

- Axioms of $\mathbf{SC}^c_{\mathcal{ALC}}$:
  - any anti-sequent

$$\Gamma_0 \dashv \Delta_0$$

  s.t. $\Gamma_0 \cap \Delta_0 = \emptyset$, where Γ_0 and Δ_0 consist of concept names only;

  - any anti-sequent of form

$$\forall r_1.\Gamma_1, \ldots, \forall r_n.\Gamma_n \dashv \exists r_1.\Delta_1, \ldots, \exists r_n.\Delta_n.$$

- Structural Rules:

$$\frac{\Gamma, C \ \dashv \ \Delta}{\Gamma \ \dashv \ \Delta}\,(w^{-1}, l) \qquad \frac{\Gamma \ \dashv \ \Delta, C}{\Gamma \ \dashv \ \Delta}\,(w^{-1}, r)$$

$$\frac{\Gamma, C \ \dashv \ \Delta}{\Gamma, C, C \ \dashv \ \Delta}\,(c^{-1}, l) \qquad \frac{\Gamma \ \dashv \ C, \Delta}{\Gamma \ \dashv \ C, C, \Delta}\,(c^{-1}, r)$$

Rejection Calculus $\mathbf{SC}^{c}_{\mathcal{ALC}}$ (ctd.)

- Propositional Rules:

$$\frac{\Gamma, C, D \dashv \Delta}{\Gamma, C \sqcap D \dashv \Delta}\,(\sqcap, l) \qquad \frac{\Gamma \dashv C, D, \Delta}{\Gamma \dashv C \sqcup D, \Delta}\,(\sqcup, r)$$

$$\frac{\Gamma, C \dashv \Delta}{\Gamma, C \sqcup D \dashv \Delta}\,(\sqcup, l)_1 \qquad \frac{\Gamma \dashv C, \Delta}{\Gamma \dashv C \sqcap D, \Delta}\,(\sqcap, r)_1$$

$$\frac{\Gamma, D \dashv \Delta}{\Gamma, C \sqcup D \dashv \Delta}\,(\sqcup, l)_2 \qquad \frac{\Gamma \dashv D, \Delta}{\Gamma \dashv C \sqcap D, \Delta}\,(\sqcap, r)_2$$

$$\frac{\Gamma \dashv C, \Delta}{\Gamma, \neg C \dashv \Delta}\,(\neg, l) \qquad \frac{\Gamma, C \dashv \Delta}{\Gamma \dashv \neg C, \Delta}\,(\neg, r)$$

$$\frac{\Gamma \dashv \Delta}{\Gamma, \top \dashv \Delta}\,(\top) \qquad \frac{\Gamma \dashv \Delta}{\Gamma \dashv \bot, \Delta}\,(\bot)$$

Rejection Calculus $\mathbf{SC}^c_{\mathcal{ALC}}$ (ctd.)

- Quantifier Rules:

$$\frac{\Gamma_0 \ \dashv\ \Delta_0 \qquad \Gamma^{r_1}, \ldots, \Gamma^{r_n} \ \dashv\ \Delta^{r_1}, \ldots, \Delta^{r_n}}{\Gamma_0, \Gamma^{r_1}, \ldots, \Gamma^{r_n} \ \dashv\ \Delta_0, \Delta^{r_1}, \ldots, \Delta^{r_n}}\ (\textsc{Mix}),$$

where $\Gamma_0 \dashv \Delta_0$ is a propositional axiom.

$$\frac{\widehat{\Gamma^{r_k}} \ \dashv\ \widetilde{\Delta^{r_k}}, C_k \ \cdots\ \widehat{\Gamma^{r_l}} \ \dashv\ \widetilde{\Delta^{r_l}}, C_l \qquad \Gamma^{r_1}, \ldots, \Gamma^{r_n} \ \dashv\ \Delta^{r_1}, \ldots, \Delta^{r_n}}{\Gamma^{r_1}, \ldots, \Gamma^{r_n} \ \dashv\ \Delta^{r_1}, \ldots, \Delta^{r_n}, \forall r_k.C_k, \ldots, \forall r_l.C_l}\ (\textsc{Mix}, \forall)$$

$$\frac{\widehat{\Gamma^{r_k}}, C_k \ \dashv\ \widetilde{\Delta^{r_k}} \ \cdots\ \widehat{\Gamma^{r_l}}, C_l \ \dashv\ \widetilde{\Delta^{r_l}} \qquad \Gamma^{r_1}, \ldots, \Gamma^{r_n} \ \dashv\ \Delta^{r_1}, \ldots, \Delta^{r_n}}{\Gamma^{r_1}, \ldots, \Gamma^{r_n}, \exists r_k.C_k, \ldots, \exists r_l.C_l \ \dashv\ \Delta^{r_1}, \ldots, \Delta^{r_n}}\ (\textsc{Mix}, \exists)$$

where $1 \leq k \leq l \leq n$.

- Notation:
  - Γ^r denotes any finite multi-set of concepts containing only concepts of form $\exists r.C$ or $\forall r.C$.
  - $\widehat{\Gamma} := \{C \mid \forall r.C \in \Gamma\}$ and $\widetilde{\Gamma} := \{C \mid \exists r.C \in \Gamma\}$.

Properties of $\mathbf{SC}^c_{\mathcal{ALC}}$

- $\mathbf{SC}^c_{\mathcal{ALC}}$ is an *analytic calculus*, i.e., it enjoys the *subformula property*.

Properties of $\mathbf{SC}^c_{\mathcal{ALC}}$

- $\mathbf{SC}^c_{\mathcal{ALC}}$ is an *analytic calculus*, i.e., it enjoys the *subformula property*.
- $\mathbf{SC}^c_{\mathcal{ALC}}$ is *sound and complete*, i.e.,
  - an anti-sequent $\Gamma \dashv \Delta$ is refutable iff it is provable in $\mathbf{SC}^c_{\mathcal{ALC}}$.

Properties of $\mathbf{SC}^c_{\mathcal{ALC}}$

- $\mathbf{SC}^c_{\mathcal{ALC}}$ is an *analytic calculus*, i.e., it enjoys the *subformula property*.
- $\mathbf{SC}^c_{\mathcal{ALC}}$ is *sound and complete*, i.e.,
  - an anti-sequent $\Gamma \dashv \Delta$ is refutable iff it is provable in $\mathbf{SC}^c_{\mathcal{ALC}}$.
- Countermodels (in the form of tree models) can be extracted from a proof in $\mathbf{SC}^c_{\mathcal{ALC}}$:
  - Assigning, from bottom to top, each anti-sequent in the proof a node of the tree.
  - The end-sequent is the root of the tree.
  - New nodes are created for each application of $(\textsc{Mix}, \forall)$ and $(\textsc{Mix}, \exists)$.
  - For an axiom $\Gamma_0 \dashv \Delta_0$ with assigned node v', we ensure that
    - $v' \in C^{\mathcal{I}}$ for each $C \in \Gamma_0$ and
    - $v' \notin D^{\mathcal{I}}$ for each $D \in \Delta_0$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$\dfrac{\dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1 \qquad \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)$$

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$\dfrac{\dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1 \qquad \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$\begin{tikzcd}
 & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & \\
C \quad v_1 & & v_2 \quad D
\end{tikzcd}$$

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$
\dfrac{
  \dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1
  \qquad
  \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}
  {\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)
}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)
$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$
\begin{tikzcd}
 & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & \\
C \quad v_1 & & v_2 \quad D
\end{tikzcd}
$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$
\dfrac{\dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1 \qquad \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\color{blue}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}}\,(\sqcap, l)
$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$
\begin{tikzcd}
 & & \color{blue}{v_0} \arrow[dl, "r"'] \arrow[dr, "r"] & & \\
C & v_1 & & v_2 & D
\end{tikzcd}
$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$
\begin{prooftree}
\AxiomC{$D \dashv C$}
\RightLabel{$(\sqcap, r)_1$}
\UnaryInfC{$D \dashv C \sqcap D$}
\AxiomC{$C \dashv D$}
\RightLabel{$(\sqcap, r)_2$}
\UnaryInfC{$C \dashv C \sqcap D$}
\AxiomC{$\dashv \exists r.(C \sqcap D)$}
\RightLabel{$(\textsc{Mix}, \exists)$}
\BinaryInfC{$\exists r.C \dashv \exists r.(C \sqcap D)$}
\RightLabel{$(\textsc{Mix}, \exists)$}
\BinaryInfC{$\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)$}
\RightLabel{$(\sqcap, l)$}
\UnaryInfC{$\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)$}
\end{prooftree}
$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$
\begin{tikzcd}
 & & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & & \\
C & v_1 & & v_2 & D
\end{tikzcd}
$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$
\dfrac{
  \dfrac{
    \dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1
    \qquad
    \dfrac{
      \dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2
      \qquad
      \dashv \exists r.(C \sqcap D)
    }{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)
  }{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)
}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)
$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$
\begin{tikzcd}
 & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & \\
C \quad v_1 & & v_2 \quad D
\end{tikzcd}
$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$
\dfrac{
  \dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1
  \qquad
  \dfrac{
    \dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2
    \qquad
    \dashv \exists r.(C \sqcap D)
  }{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)
}{
  \dfrac{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)
}\,(\textsc{Mix}, \exists)
$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$
\begin{tikzcd}
 & & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & & \\
C & v_1 & & v_2 & D
\end{tikzcd}
$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$\dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1 \qquad \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\textsc{Mix}, \exists)}{\dfrac{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)}\,(\textsc{Mix}, \exists)$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$\begin{tikzcd} & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & \\ C \quad v_1 & & v_2 \quad D \end{tikzcd}$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Example: Extracting a Counter Model

We refute $\exists r.C \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

$$\dfrac{\dfrac{\dfrac{D \dashv C}{D \dashv C \sqcap D}\,(\sqcap, r)_1 \qquad \dfrac{\dfrac{C \dashv D}{C \dashv C \sqcap D}\,(\sqcap, r)_2 \qquad \dashv \exists r.(C \sqcap D)}{\exists r.C \dashv \exists r.(C \sqcap D)}\,(\mathrm{M{\scriptsize IX}}, \exists)}{\exists r.C, \exists r.D \dashv \exists r.(C \sqcap D)}\,(\mathrm{M{\scriptsize IX}}, \exists)}{\exists r.C \sqcap \exists r.D \dashv \exists r.(C \sqcap D)}\,(\sqcap, l)$$

$\mathcal{I} = \langle \{v_0, v_1, v_2\}, \cdot^{\mathcal{I}} \rangle$, $r^{\mathcal{I}} = \{(v_0, v_1), (v_0, v_2)\}$, $C^{\mathcal{I}} = \{v_1\}$, $D^{\mathcal{I}} = \{v_2\}$.

$$\begin{tikzcd} & v_0 \arrow[dl, "r"'] \arrow[dr, "r"] & \\ C \quad v_1 & & v_2 \quad D \end{tikzcd}$$

$\Longrightarrow$ $(\exists r.C \sqcap \exists r.D)^{\mathcal{I}} = \{v_0\}$ but $(\exists r.(C \sqcap D))^{\mathcal{I}} = \emptyset$.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.
- Well known algorithm for $\mathcal{ALC}$ (Baader & Sattler, 2001) is based on the notion of a *completion graph*.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.
- Well known algorithm for $\mathcal{ALC}$ (Baader & Sattler, 2001) is based on the notion of a *completion graph*.
  - A model can be extracted from a complete completion graph.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.
- Well known algorithm for $\mathcal{ALC}$ (Baader & Sattler, 2001) is based on the notion of a *completion graph*.
  - A model can be extracted from a complete completion graph.
- A complete completion graph can be obtained from a proof in $\mathbf{SC}^c_{\mathcal{ALC}}$.

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.
- Well known algorithm for $\mathcal{ALC}$ (Baader & Sattler, 2001) is based on the notion of a *completion graph*.
  - A model can be extracted from a complete completion graph.
- A complete completion graph can be obtained from a proof in $\mathbf{SC}^c_{\mathcal{ALC}}$.
- ☞ Although tableaux are usually syntactic variants of standard Gentzen systems, in the case of description logics, tableaux axiomatise *satisfiability*

Relation to Tableau Algorithms

- Tableau algorithms are the most common reasoning procedures for description logics.
- They rely on the construction of a *canonical model* which witnesses the satisfiability of a concept or a knowledge base.
- Well known algorithm for $\mathcal{ALC}$ (Baader & Sattler, 2001) is based on the notion of a *completion graph*.
  - A model can be extracted from a complete completion graph.
- A complete completion graph can be obtained from a proof in $\mathbf{SC}^c_{\mathcal{ALC}}$.
- ☞ Although tableaux are usually syntactic variants of standard Gentzen systems, in the case of description logics, tableaux axiomatise *satisfiability*
  - they therefore correspond to a rejection calculus.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$

- $\mathcal{ALC}$ can be translated into a multi-modal version of the modal logic $\mathbf{K}$.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$

- $\mathcal{ALC}$ can be translated into a multi-modal version of the modal logic $\mathbf{K}$.
- Based on this translation, we obtain a rejection calculus for multi-modal $\mathbf{K}$, generalising a rejection calculus for standard $\mathbf{K}$ by Goranko (1994).

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$

- $\mathcal{ALC}$ can be translated into a multi-modal version of the modal logic $\mathbf{K}$.
- Based on this translation, we obtain a rejection calculus for multi-modal $\mathbf{K}$, generalising a rejection calculus for standard $\mathbf{K}$ by Goranko (1994).
- In multi-modal $\mathbf{K}$, we have different modal operators $[\alpha]$, where α is a modality.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$

- $\mathcal{ALC}$ can be translated into a multi-modal version of the modal logic $\mathbf{K}$.
- Based on this translation, we obtain a rejection calculus for multi-modal $\mathbf{K}$, generalising a rejection calculus for standard $\mathbf{K}$ by Goranko (1994).
- In multi-modal $\mathbf{K}$, we have different modal operators $[\alpha]$, where α is a modality.
  - Semantically, multi-modal $\mathbf{K}$ is based on Kripke models $\mathcal{M} = \langle W, \{R_\alpha\}_{\alpha \in \tau}, V \rangle$, where
    - W is a non-empty set of *worlds*,
    - $R_\alpha \subseteq W \times W$ defines an *accessibility relation* for each modality α, and
    - V defines which propositional variables are true at which worlds.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$ (ctd.)

- The translation of $\mathcal{ALC}$ into multi-modal $\mathbf{K}$ is simply by viewing concepts of form $\forall r.C$ as modal formulae of form $[\alpha]C'$,
  - C' is the corresponding translation of the concept C,
  - each role name corresponds to one and only one modality.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$ (ctd.)

- The translation of $\mathcal{ALC}$ into multi-modal $\mathbf{K}$ is simply by viewing concepts of form $\forall r.C$ as modal formulae of form $[\alpha]C'$,
  - C' is the corresponding translation of the concept C,
  - each role name corresponds to one and only one modality.
- Based on this translation, we can directly translate our calculus into corresponding modal rules.

A Rejection Calculus for Multi-Modal Logic $\mathbf{K}$ (ctd.)

- The translation of $\mathcal{ALC}$ into multi-modal $\mathbf{K}$ is simply by viewing concepts of form $\forall r.C$ as modal formulae of form $[\alpha]C'$,
  - C' is the corresponding translation of the concept C,
  - each role name corresponds to one and only one modality.
- Based on this translation, we can directly translate our calculus into corresponding modal rules.
- For instance, the rule $(\textsc{Mix})$ becomes

$$\frac{\Gamma_0 \ \dashv \ \Delta_0 \qquad [\alpha_1]\Gamma_1, \ldots, [\alpha_n]\Gamma_n \ \dashv \ [\alpha_1]\Delta_1, \ldots, [\alpha_n]\Delta_n}{\Gamma_0, [\alpha_1]\Gamma_1, \ldots, [\alpha_n]\Gamma_n \ \dashv \ \Delta_0, [\alpha_1]\Delta_1, \ldots, [\alpha_n]\Delta_n} \ (\textsc{Mix})$$

  where

  - Γ_0, Δ_0 are disjoint sets of propositional variables and
  - $[\alpha]\Gamma := \{[\alpha]\varphi \mid \varphi \in \Gamma\}$.

Conclusion

- We presented a sound and complete rejection system axiomatising non-subsumption in $\mathcal{ALC}$.
  - Rejection proofs are witnesses for tree-like counterexamples.

Conclusion

- We presented a sound and complete rejection system axiomatising non-subsumption in $\mathcal{ALC}$.
  - Rejection proofs are witnesses for tree-like counterexamples.
- Future work:
  - Provide a calculus for the general case of dealing with *reasoning from knowledge bases*, i.e., taking TBox reasoning into account.

Conclusion

- We presented a sound and complete rejection system axiomatising non-subsumption in $\mathcal{ALC}$.
  - Rejection proofs are witnesses for tree-like counterexamples.
- Future work:
  - Provide a calculus for the general case of dealing with *reasoning from knowledge bases*, i.e., taking TBox reasoning into account.
  - Study calculi for *nonmonotonic extensions* of description logics.

Conclusion

- We presented a sound and complete rejection system axiomatising non-subsumption in $\mathcal{ALC}$.
  - Rejection proofs are witnesses for tree-like counterexamples.
- Future work:
  - Provide a calculus for the general case of dealing with *reasoning from knowledge bases*, i.e., taking TBox reasoning into account.
  - Study calculi for *nonmonotonic extensions* of description logics.
  - Can a combination of traditional proof methods and rejection calculi yield potentially more efficient reasoning systems?

Conclusion

- We presented a sound and complete rejection system axiomatising non-subsumption in $\mathcal{ALC}$.
  - Rejection proofs are witnesses for tree-like counterexamples.
- Future work:
  - Provide a calculus for the general case of dealing with *reasoning from knowledge bases*, i.e., taking TBox reasoning into account.
  - Study calculi for *nonmonotonic extensions* of description logics.
  - Can a combination of traditional proof methods and rejection calculi yield potentially more efficient reasoning systems?
    - E.g., by effects of simulating versions of the cut rule.