A General Tableau Method for Deciding Description Logics, Modal Logics and Related First-Order Fragments

Dmitry Tishkovsky

joint work with
Renate A. Schmidt

School of Computer Science
The University of Manchester
dmitry.tishkovsky@manchester.ac.uk

PRAGUE, CZECH REPUBLIC

11 JUNE 2008

Outline

Increasing Demand for Reasoning Tools

- Description logics form a basis for web ontology languages, OWL DL and OWL 1.1
- Modal and dynamic logics are useful in multi-agent reasoning
- Metric logics are intended to be helpful in classification problems
- Fuzzy logics . . . , etc

Reasoning tools are of increased demand.

Increasing Demand for Reasoning Tools

- Description logics form a basis for web ontology languages, OWL DL and OWL 1.1
- Modal and dynamic logics are useful in multi-agent reasoning
- Metric logics are intended to be helpful in classification problems
- Fuzzy logics . . . , etc

Reasoning tools are of increased demand.

Increasing Demand for Reasoning Tools

- Description logics form a basis for web ontology languages, OWL DL and OWL 1.1
- Modal and dynamic logics are useful in multi-agent reasoning
- Metric logics are intended to be helpful in classification problems
- Fuzzy logics . . . , etc

Reasoning tools are of increased demand.

MANCHESTER 1824 The University of Manchester

Increasing Demand for Reasoning Tools

- Description logics form a basis for web ontology languages, OWL DL and OWL 1.1
- Modal and dynamic logics are useful in multi-agent reasoning
- Metric logics are intended to be helpful in classification problems
- Fuzzy logics . . . , etc

Reasoning tools are of increased demand.

Increasing Demand for Reasoning Tools

- Description logics form a basis for web ontology languages, OWL DL and OWL 1.1
- Modal and dynamic logics are useful in multi-agent reasoning
- Metric logics are intended to be helpful in classification problems
- Fuzzy logics . . . , etc

Reasoning tools are of increased demand.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

MANCHESTER 1824 The University of Manchester

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
- Tableau prover engineering platforms: LWB, TWB, LoTREC, ...
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, . . .
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, . . .
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, . . .
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LoTREC, . . .
- Other: KAON2, . . .

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FaCT++, RacerPro, Pellet, DLP, …
  are highly optimised but not generic
- Resolution provers: MSpass, Vampire, …
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: Isabelle, Coq, …
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LoTREC, …
  allow to play with rules but do not always ensure termination
- Other: KAON2, …

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
  allow to play with rules but do not always ensure termination
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
  allow to play with rules but do not always ensure termination
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

MANCHESTER 1824 The University of Manchester

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, . . .
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, . . .
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, . . .
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, . . .
  allow to play with rules but do not always ensure termination
- Other: KAON2, . . .

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
  allow to play with rules but do not always ensure termination
- Other: KAON2, ...
  ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
  allow to play with rules but do not always ensure termination
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

MANCHESTER 1824 The University of Manchester

Reasoning Tools

- Tableau provers: FACT++, RACERPRO, PELLET, DLP, ...
  are highly optimised but not generic
- Resolution provers: MSPASS, VAMPIRE, ...
  are difficult to tune to decide a particular logic (a first-order fragment)
- Generic interactive platforms: ISABELLE, COQ, ...
  do not provide automated decision procedures
- Tableau prover engineering platforms: LWB, TWB, LOTREC, ...
  allow to play with rules but do not always ensure termination
- Other: KAON2, ...

Answer to the increased demand for decision procedures for logical systems from existing automated reasoning tools is not sufficient.

Our Approach
Tableau Prover Synthesis

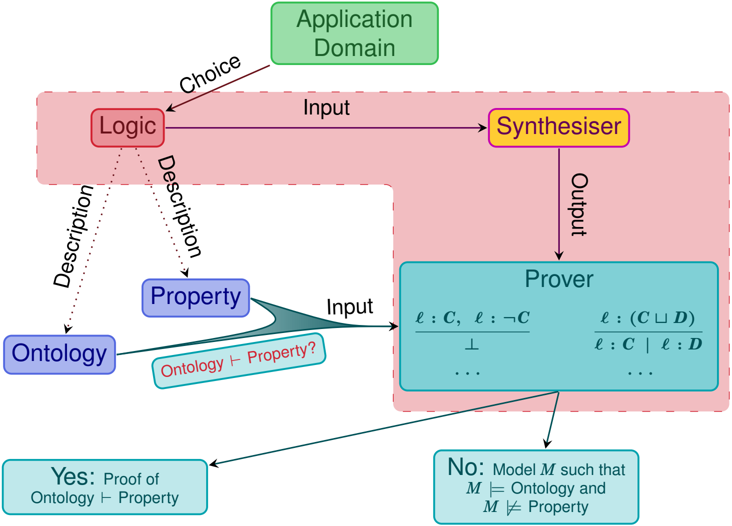


Tableau Termination Problem

- How to ensure termination of a tableau algorithm?
- An appropriate blocking mechanism is needed, e.g.:
  - subset or equality blocking,
  - dynamic or static blocking,
  - successor or anywhere blocking,
  - combinations of the above.
- Problem: How to
  - define a general blocking mechanism which unifies all the standard ones and
  - describe a class of logics for which the general blocking mechanism ensures termination of corresponding tableau algorithms?

Tableau Termination Problem

- How to ensure termination of a tableau algorithm?
- An appropriate blocking mechanism is needed, e.g.:
  - subset or equality blocking,
  - dynamic or static blocking,
  - successor or anywhere blocking,
  - combinations of the above.
- Problem: How to
  - define a general blocking mechanism which unifies all the standard ones and
  - describe a class of logics for which the general blocking mechanism ensures termination of corresponding tableau algorithms?

MANCHESTER 1824 The University of Manchester

Tableau Termination Problem

- How to ensure termination of a tableau algorithm?
- An appropriate blocking mechanism is needed, e.g.:
  - subset or equality blocking,
  - dynamic or static blocking,
  - successor or anywhere blocking,
  - combinations of the above.
- Problem: How to
  - define a general blocking mechanism which unifies all the standard ones and
  - describe a class of logics for which the general blocking mechanism ensures termination of corresponding tableau algorithms?

Motivation Summary

- Absence of general decision procedures in automated reasoning for tableaux and instantiation-based methods.
- Absence of a theoretical foundations for generic platforms in which tableau decision procedures can be built in a uniform way for different logics and different applications.
- The work is based on observation that proofs of termination of tableau algorithms and proofs of the effective finite model property by the filtration argument are very similar.

Motivation Summary

- Absence of general decision procedures in automated reasoning for tableaux and instantiation-based methods.
- Absence of a theoretical foundations for generic platforms in which tableau decision procedures can be built in a uniform way for different logics and different applications.
- The work is based on observation that proofs of termination of tableau algorithms and proofs of the effective finite model property by the filtration argument are very similar.

MANCHESTER 1824 The University of Manchester

Motivation Summary

- Absence of general decision procedures in automated reasoning for tableaux and instantiation-based methods.
- Absence of a theoretical foundations for generic platforms in which tableau decision procedures can be built in a uniform way for different logics and different applications.
- The work is based on observation that proofs of termination of tableau algorithms and proofs of the effective finite model property by the filtration argument are very similar.

Outline

Syntax and Semantics

individual

Concepts: $C, D \stackrel{\text{def}}{=} p \mid \neg C \mid C \sqcup D \mid \exists R.C \mid \{\ell\} \mid \ell : C$

Roles: $R, R_i \stackrel{\text{def}}{=} r \mid \rho_0(R_1, \ldots, R_{\mu_0}) \mid \rho_1(R_1, \ldots, R_{\mu_1}) \mid \ldots$

MANCHESTER 1824 The University of Manchester

Syntax and Semantics

individual

Concepts: $C, D \stackrel{\text{def}}{=} p \mid \neg C \mid C \sqcup D \mid \exists R.C \mid \{\ell\} \mid \ell : C$

Roles: $R, R_i \stackrel{\text{def}}{=} r \mid \rho_0(R_1, \ldots, R_{\mu_0}) \mid \rho_1(R_1, \ldots, R_{\mu_1}) \mid \ldots$

Interpretation (model): $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfying

$$p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \qquad r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \qquad \ell^{\mathcal{I}} \in \Delta^{\mathcal{I}}$$

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y \in C^{\mathcal{I}}\ (x, y) \in R^{\mathcal{I}}\} \qquad (\{\ell\})^{\mathcal{I}} = \{\ell^{\mathcal{I}}\}$$

$$(\ell : C)^{\mathcal{I}} = \begin{cases} \Delta^{\mathcal{I}}, & \text{if } \ell^{\mathcal{I}} \in C^{\mathcal{I}}, \\ \varnothing, & \text{otherwise, and} \end{cases}$$

additional semantic conditions for $\rho_0, \rho_1, \ldots$

.

Example: $\mathcal{SO}$ — Logic with Transitive Roles

- Language extended by transitive role constants $s \in$ Trans.
- For every $s \in$ Trans and a model $\mathcal{I}$, the interpretation of $s^{\mathcal{I}}$ is a transitive relation on $\mathcal{I}$:
  $(x, y), (y, z) \in s^{\mathcal{I}}$ implies $(x, z) \in s^{\mathcal{I}}$ for all $x, y, z \in \Delta^{\mathcal{I}}$.

Example: $\mathcal{ALBO}$ — Logic with Boolean Role Operators

- Extra operators on roles: role inverse R^{-1}, role complement $\neg R$, and role union $R \sqcup S$.
- Interpretations of the operators:

$$(\neg R)^{\mathcal{I}} \stackrel{\text{def}}{=} (\Delta \times \Delta) \setminus R^{\mathcal{I}}$$
$$(R \sqcup S)^{\mathcal{I}} \stackrel{\text{def}}{=} R^{\mathcal{I}} \cup S^{\mathcal{I}}$$
$$(R^{-1})^{\mathcal{I}} \stackrel{\text{def}}{=} (R^{\mathcal{I}})^{-1} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$$

Properties

- $\mathcal{ALBO}$ is out of the mainstream DLs.
- $\mathcal{ALBO}$ subsumes two variable fragment of first-order logic.
- $\mathcal{ALBO}$ is decidable by resolution.
- Satisfiability problem for $\mathcal{ALBO}$ is NExpTime-complete.
- Very expressive: universal modality and Boolean combinations of role inclusions $R \sqsubseteq S$, concept inclusions $C \sqsubseteq D$, concept assertions $\ell : C$, role assertions $(\ell, \ell') : D$, etc are expressible in $\mathcal{ALBO}$.

MANCHESTER 1824 The University of Manchester

Example: $\mathcal{ALBO}$ — Logic with Boolean Role Operators

- Extra operators on roles: role inverse R^{-1}, role complement $\neg R$, and role union $R \sqcup S$.
- Interpretations of the operators:

$$(\neg R)^{\mathcal{I}} \stackrel{\text{def}}{=} (\Delta \times \Delta) \setminus R^{\mathcal{I}}$$
$$(R \sqcup S)^{\mathcal{I}} \stackrel{\text{def}}{=} R^{\mathcal{I}} \cup S^{\mathcal{I}}$$
$$(R^{-1})^{\mathcal{I}} \stackrel{\text{def}}{=} (R^{\mathcal{I}})^{-1} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$$

Properties

- $\mathcal{ALBO}$ is out of the mainstream DLs.
- $\mathcal{ALBO}$ subsumes two variable fragment of first-order logic.
- $\mathcal{ALBO}$ is decidable by resolution.
- Satisfiability problem for $\mathcal{ALBO}$ is NExpTime-complete.
- Very expressive: universal modality and Boolean combinations of role inclusions $R \sqsubseteq S$, concept inclusions $C \sqsubseteq D$, concept assertions $\ell : C$, role assertions $(\ell, \ell') : D$, etc are expressible in $\mathcal{ALBO}$.

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \text{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\text{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \text{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\text{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \mathrm{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\mathrm{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \mathrm{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\mathrm{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \text{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\text{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \text{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\text{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \text{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\text{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \text{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\text{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

MANCHESTER 1824 The University of Manchester

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \text{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\text{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \text{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\text{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

MANCHESTER 1824 The University of Manchester

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq$ sub(Σ) for every Σ.
- sub is *finite* iff sub(Σ) is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma =$ sub(Σ).
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. sub(Σ) is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

Closure operator

- sub is a monotone operator on sets of expressions.
- $\Sigma \subseteq \text{sub}(\Sigma)$ for every Σ.
- sub is *finite* iff $\text{sub}(\Sigma)$ is finite whenever Σ is finite.
- A finite sub can be replaced by an equivalent notion of a well-founded ordering on expressions.
- Σ is sub-*closed*, or a *signature* iff $\Sigma = \text{sub}(\Sigma)$.
- Usually, there is a lot of flexibility in choice of sub.

Example

- sub for $\mathcal{SO}$ and $\mathcal{ALBO}$ can be chosen as the subexpression operator, i.e. $\text{sub}(\Sigma)$ is a set of all subexpressions of expressions in Σ.
- sub for *PDL* includes more expressions.

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
- *L admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x],[y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x\, \exists y' \sim y\, (x',y') \in R^{\mathcal{I}}$
- L *admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x],[y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x \exists y' \sim y\, (x', y') \in R^{\mathcal{I}}$
- *L admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x],[y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x \exists y' \sim y\, (x', y') \in R^{\mathcal{I}}$
- L *admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, \cdot^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x],[y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x \exists y' \sim y\ (x', y') \in R^{\mathcal{I}}$
- L *admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

MANCHESTER 1824 The University of Manchester

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x], [y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x \exists y' \sim y \, (x', y') \in R^{\mathcal{I}}$
- L *admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

Filtration

- $\mathcal{I}$ is a model
- $\sim$ is an equivalence relation on $\Delta^{\mathcal{I}}$
- $[x] \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$
- *Filtration* of $\mathcal{I}$ is a structure $\overline{\mathcal{I}} = (\Delta^{\overline{\mathcal{I}}}, .^{\overline{\mathcal{I}}})$ such that
  - $\Delta^{\overline{\mathcal{I}}} = \{[x] \mid x \in \Delta^{\mathcal{I}}\}$,
  - $C^{\overline{\mathcal{I}}} = \{[x] \mid x \in C^{\mathcal{I}}\}$,
  - $\ell^{\overline{\mathcal{I}}} = [\ell^{\mathcal{I}}]$, and
  - $([x],[y]) \in R^{\overline{\mathcal{I}}}$ whenever $\exists x' \sim x \exists y' \sim y\ (x',y') \in R^{\mathcal{I}}$
- L *admits finite filtration* iff for every finite L-signature Σ and every L-model $\mathcal{I}$ of the signature Σ there exists an equivalence relation $\sim$ on $\mathcal{I}$ such that there is a $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ which is a finite L-model of the signature Σ.

Theorem

Let L be a logic and sub *be a finite expression closure operator. If L admits finite filtration then L has the effective finite model property.*

$\mathcal{SO}$ Filtration

- Given an $\mathcal{SO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x) \stackrel{\text{def}}{=} \{C \in \Sigma \mid x \in C^{\mathcal{I}}\}.$$

- The equivalence $\sim$ defined by

$$x \sim y \stackrel{\text{def}}{\Longleftrightarrow} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

  for every $x, y \in \Delta^{\mathcal{I}}$.
- An interpretation of every role r in the $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ is defined by

$$r^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{([x],[y]) \mid y \in C^{\mathcal{I}} \text{ implies } x \in (\exists r.C)^{\mathcal{I}} \text{ for every } \exists r.C \in \Sigma\}.$$

$\mathcal{SO}$ Filtration

- Given an $\mathcal{SO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x) \stackrel{\text{def}}{=} \{C \in \Sigma \mid x \in C^{\mathcal{I}}\}.$$

- The equivalence $\sim$ defined by

$$x \sim y \stackrel{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

for every $x, y \in \Delta^{\mathcal{I}}$.

- An interpretation of every role r in the $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ is defined by

$$r^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{([x],[y]) \mid y \in C^{\mathcal{I}} \text{ implies } x \in (\exists r.C)^{\mathcal{I}} \text{ for every } \exists r.C \in \Sigma\}.$$

MANCHESTER 1824 The University of Manchester

$\mathcal{SO}$ Filtration

- Given an $\mathcal{SO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x) \stackrel{\text{def}}{=} \{C \in \Sigma \mid x \in C^{\mathcal{I}}\}.$$

- The equivalence $\sim$ defined by

$$x \sim y \stackrel{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

  for every $x, y \in \Delta^{\mathcal{I}}$.

- An interpretation of every role r in the $\sim$-filtration $\overline{\mathcal{I}}$ of $\mathcal{I}$ is defined by

$$r^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{([x],[y]) \mid y \in C^{\mathcal{I}} \text{ implies } x \in (\exists r.C)^{\mathcal{I}} \text{ for every } \exists r.C \in \Sigma\}.$$

MANCHESTER 1824 The University of Manchester

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y \, (x', y') \in R^{\mathcal{I}}\}$.
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \Longrightarrow \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$

$$x \sim x' \wedge y \sim y' \Longrightarrow \tau^{\Sigma}(x,x') = \tau^{\Sigma}(y,y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.

MANCHESTER 1824 The University of Manchester

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\Longleftrightarrow} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y\ (x', y') \in R^{\mathcal{I}}\}$.
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \Longrightarrow \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$

$$x \sim x' \wedge y \sim y' \Longrightarrow \tau^{\Sigma}(x, x') = \tau^{\Sigma}(y, y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\Longleftrightarrow} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y\, (x', y') \in R^{\mathcal{I}}\}$.
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \Longrightarrow \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$

$$x \sim x' \wedge y \sim y' \Longrightarrow \tau^{\Sigma}(x,x') = \tau^{\Sigma}(y,y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.
  - It always produces an $\mathcal{ALBO}$-model but the problem is to make it finite.

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y\, (x', y') \in R^{\mathcal{I}}\}$.
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \Longrightarrow \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$

$$x \sim x' \wedge y \sim y' \Longrightarrow \tau^{\Sigma}(x,x') = \tau^{\Sigma}(y,y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.
  - It always produces an $\mathcal{ALBO}$-model but the problem is to make it finite.

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y\, (x', y') \in R^{\mathcal{I}}\}$.
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \implies \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$
$$x \sim x' \wedge y \sim y' \implies \tau^{\Sigma}(x,x') = \tau^{\Sigma}(y,y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.
  - It always produces an $\mathcal{ALBO}$-model but the problem is to make it finite.

MANCHESTER 1824 The University of Manchester

$\mathcal{ALBO}$ Filtration

- Given an $\mathcal{ALBO}$-model $\mathcal{I}$ and a signature Σ, let

$$\tau^{\Sigma}(x,y) \stackrel{\text{def}}{=} \{R \in \Sigma \mid (x,y) \in R^{\mathcal{I}}\}.$$

- Standard filtration:
  - The equivalence $\sim$ defined by

$$x \simeq y \stackrel{\text{def}}{\Longleftrightarrow} \tau^{\Sigma}(x) = \tau^{\Sigma}(y)$$

    for every $x, y \in \Delta^{\mathcal{I}}$.
  - $R^{\overline{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\lfloor x \rfloor, \lfloor y \rfloor) \mid \exists x' \simeq x \exists y' \simeq y\, (x', y') \in R^{\mathcal{I}}\}.$
  - It is finite but, in general, does not produce an $\mathcal{ALBO}$-model: the property $(\neg R)^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$ is affected.
- Nice filtration:
  - The equivalence $\sim$ satisfies

$$x \sim y \Longrightarrow \tau^{\Sigma}(x) = \tau^{\Sigma}(y),$$

$$x \sim x' \wedge y \sim y' \Longrightarrow \tau^{\Sigma}(x, x') = \tau^{\Sigma}(y, y')$$

    for every $x, y, x', y' \in \Delta^{\mathcal{I}}$.
  - It always produces an $\mathcal{ALBO}$-model but the problem is to make it finite.

MANCHESTER 1824 The University of Manchester

Example of a Nice $\mathcal{ALBO}$ Filtration

$$x \cong y \overset{\text{def}}{\Longleftrightarrow} \tau^{\Sigma}(x) = \tau^{\Sigma}(y) \text{ and }$$
$$\tau^{\Sigma}(x,z) = \tau^{\Sigma}(y,z) \text{ and } \tau^{\Sigma}(z,x) = \tau^{\Sigma}(z,y) \text{ for all } z \in \Delta^{\mathcal{I}}.$$

It is not finite!

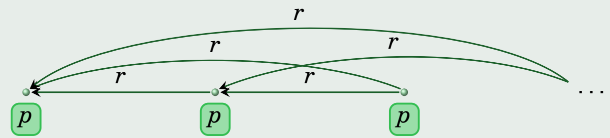
Counterexample

Example of a Nice $\mathcal{ALBO}$ Filtration

$$x \cong y \overset{\text{def}}{\iff} \tau^{\Sigma}(x) = \tau^{\Sigma}(y) \text{ and } \tau^{\Sigma}(x,z) = \tau^{\Sigma}(y,z) \text{ and } \tau^{\Sigma}(z,x) = \tau^{\Sigma}(z,y) \text{ for all } z \in \Delta^{\mathcal{I}}.$$

It is not finite!

Counterexample



Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
  - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
  - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
  - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
  - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
  - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
  - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

MANCHESTER 1824 The University of Manchester

Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
  - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
  - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
  - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
  - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

Conflict Elimination

- Introduced by Gargov, Passy, and Tinchev for *BML*.
- Works for *BML* and $\mathcal{ALB}$ but, in general, fails if individuals are in the language.
- *Quasi-model*: $\mathcal{I}$ where (possibly) $(\neg R)^{\mathcal{I}} \not\subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$.
- If $\mathcal{ALB}$-quasi-model $\mathcal{I}$ is finite and Σ is a finite signature then there are
    - a finite $\mathcal{ALB}$-model $\mathcal{I}'$ and
    - a p-morphism f (w.r.t. Σ) from $\mathcal{I}'$ onto $\mathcal{I}$.

Theorem

If a $\mathcal{ALB}$-concept C is satisfiable in a quasi-model then it is satisfiable in a finite model.

Corollary

- *$\mathcal{ALB}$ is complete with respect to the class of all $\mathcal{ALB}$-quasi-models.*
- *$\mathcal{ALB}$ admits finite filtration over the class of all $\mathcal{ALB}$-quasi-models.*

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\iff} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u, z) = \tau^{\Sigma}(u, z) \text{ and } \tau^{\Sigma}(z, u) = \tau^{\Sigma}(z, u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\underline{\mathcal{I}}'$:

$$x \sim y \stackrel{\text{def}}{\iff} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = [x] = [y], \quad \tau^{\Sigma}(u,z) = \tau^{\Sigma}(u,z) \text{ and } \tau^{\Sigma}(z,u) = \tau^{\Sigma}(z,u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\underline{\mathcal{I}}'$ and apply the defined nice filtration to the original $\underline{\mathcal{I}}$.

Theorem

- *$\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.*
- *$\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.*
- *$\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).*

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \overset{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \overset{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \overset{\text{def}}{\iff} x \simeq y \text{ and for all } u,z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u,z) = \tau^{\Sigma}(u,z) \text{ and } \tau^{\Sigma}(z,u) = \tau^{\Sigma}(z,u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ.
  Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$.
  Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\Longleftrightarrow} x \simeq y \text{ and for all } u, z \in \Delta^{\mathcal{I}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u,z) = \tau^{\Sigma}(u,z) \text{ and } \tau^{\Sigma}(z,u) = \tau^{\Sigma}(z,u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ.
  Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$.
  Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\Longleftrightarrow} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u, z) = \tau^{\Sigma}(u, z) \text{ and } \tau^{\Sigma}(z, u) = \tau^{\Sigma}(z, u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \overset{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \overset{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \overset{\text{def}}{\iff} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u,z) = \tau^{\Sigma}(u,z) \text{ and } \tau^{\Sigma}(z,u) = \tau^{\Sigma}(z,u).$$

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\iff} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u,z) = \tau^{\Sigma}(u,z) \text{ and } \tau^{\Sigma}(z,u) = \tau^{\Sigma}(z,u).$$

This filtration is finite because $\underline{\mathcal{I}}'$ is finite!

- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ. Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$. Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\Longleftrightarrow} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u, z) = \tau^{\Sigma}(u, z) \text{ and } \tau^{\Sigma}(z, u) = \tau^{\Sigma}(z, u).$$

  This filtration is finite because $\underline{\mathcal{I}}'$ is finite!
- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- $\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.
- $\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).

A Finite Nice $\mathcal{ALBO}$ Filtration

- Take an $\mathcal{ALBO}$-concept C and an $\mathcal{ALBO}$-model $\mathcal{I}$ satisfying C.
- Replace all singleton subconcepts $\{\ell\}$ in C by fresh propositional symbols p_ℓ.
  Let C' be the result of the replacement and $\Sigma \stackrel{\text{def}}{=} \text{sub}(C')$.
- Make an $\mathcal{ALB}$-model $\mathcal{I}'$ from $\mathcal{I}$ by making interpretation $p_\ell^{\mathcal{I}'} \stackrel{\text{def}}{=} \{\ell\}^{\mathcal{I}}$.
  Clearly, $\mathcal{I}'$ satisfies C'.
- Obtain a finite $\mathcal{ALB}$-quasi-model $\underline{\mathcal{I}}$ satisfying C using the standard filtration on $\mathcal{I}'$.
- Obtain (by the process of conflict elimination) an $\mathcal{ALB}$-model $\underline{\mathcal{I}}'$ and a p-morphism f from $\underline{\mathcal{I}}'$ onto $\underline{\mathcal{I}}$.
- Having $\underline{\mathcal{I}}'$ in hand, define a nice filtration on $\mathcal{I}'$:

$$x \sim y \stackrel{\text{def}}{\iff} x \simeq y \text{ and for all } u, z \in \Delta^{\underline{\mathcal{I}}'} \text{ such that } f(u) = \lfloor x \rfloor = \lfloor y \rfloor,$$
$$\tau^{\Sigma}(u, z) = \tau^{\Sigma}(u, z) \text{ and } \tau^{\Sigma}(z, u) = \tau^{\Sigma}(z, u).$$

  This filtration is finite because $\underline{\mathcal{I}}'$ is finite!
- Replace p_ℓ back for $\{\ell\}$ in C', Σ, and $\mathcal{I}'$ and apply the defined nice filtration to the original $\mathcal{I}$.

Theorem

- *$\mathcal{ALBO}$ is complete with respect to the class of all $\mathcal{ALBO}$-quasi-models.*
- *$\mathcal{ALBO}$ admits finite (standard) filtration over the class of all $\mathcal{ALBO}$-quasi-models.*
- *$\mathcal{ALBO}$ admits finite (nice) filtration (over the class of all $\mathcal{ALBO}$-models).*

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \dots, \quad \ell_n : C_n}{\ell^1_1 : D^1_1, \quad \dots, \quad \ell^1_{k_1} : D^1_{k_1} \mid \cdots \mid \ell^m_1 : D^m_1, \quad \dots, \quad \ell^m_{k_m} : D^m_{k_m}}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \ \ldots, \ \ell_n : C_n}{\ell_1^1 : D_1^1, \ \ldots, \ \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \ \ldots, \ \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \ \ldots, \ \ell_n : C_n}{\ell_1^1 : D_1^1, \ \ldots, \ \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \ \ldots, \ \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \dots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \dots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \dots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \ \ldots, \ \ell_n : C_n}{\ell_1^1 : D_1^1, \ \ldots, \ \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \ \ldots, \ \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Tableau Calculus

- Tableau rule: $\dfrac{\ell_1 : C_1, \quad \ldots, \quad \ell_n : C_n}{\ell_1^1 : D_1^1, \quad \ldots, \quad \ell_{k_1}^1 : D_{k_1}^1 \mid \cdots \mid \ell_1^m : D_1^m, \quad \ldots, \quad \ell_{k_m}^m : D_{k_m}^m}$.
- A *clash* rule is a tableau rule where $m = 0$.
- Tableau calculus T is a set of tableau rules.
- Given a concept C, tableau $T(C)$ is a (completely) expanded tree of sets of concepts such that
  - the root node is $\{\ell : C\}$ for some fresh individual ℓ;
  - every child node is obtained by application of some T-rule to concepts from the parent node.
- A branch of $T(C)$ is *closed* if a clash rule is applied in it. A branch is *open* if it is not closed.
- $T(C)$ is closed if all its branches are closed, and it is open if there is an open branch in it.
- T is sound for a logic L if $T(C)$ is open for every L-satisfiable concept C.
- T is *complete* for L if C has an L-model whenever $T(C)$ is open.
- T is *terminating* for L if every open T-tableau contains a *finite* open branch.

Common Tableau Rules

Standard rules for $\mathcal{ALC}$

$$(\bot)\ \frac{\ell : C, \quad \ell : \neg C}{\bot} \qquad (\neg\neg)\ \frac{\ell : \neg\neg C}{\ell : C}$$

$$(\neg\sqcup)\ \frac{\ell : \neg(C \sqcup D)}{\ell : \neg C, \quad \ell : \neg D} \qquad (\sqcup)\ \frac{\ell : (C \sqcup D)}{\ell : C \mid \ell : D}$$

$$(\exists)\ \frac{\ell : \exists R.C}{\ell : \exists R.\{\ell'\}, \quad \ell' : C}(\ell' \text{ is new}) \qquad (\neg\exists)\ \frac{\ell : \neg\exists R.C, \quad \ell : \exists R.\{\ell'\}}{\ell' : \neg C}$$

Rules for individuals

$$(\text{sym})\ \frac{\ell : \{\ell'\}}{\ell' : \{\ell\}} \qquad (\neg\text{sym})\ \frac{\ell : \neg\{\ell'\}}{\ell' : \neg\{\ell\}} \qquad (\text{ref})\ \frac{\ell : C}{\ell : \{\ell\}}$$

$$(\text{mon})\ \frac{\ell : \{\ell'\}, \quad \ell' : C}{\ell : C} \qquad (\text{canc})\ \frac{\ell : (\ell' : C)}{\ell' : C}$$

MANCHESTER 1824 The University of Manchester

Common Tableau Rules

Standard rules for $\mathcal{ALC}$

$(\bot)\ \dfrac{\ell : C,\ \ \ell : \neg C}{\bot}$

$(\neg\neg)\ \dfrac{\ell : \neg\neg C}{\ell : C}$

$(\neg\sqcup)\ \dfrac{\ell : \neg(C \sqcup D)}{\ell : \neg C,\ \ \ell : \neg D}$

$(\sqcup)\ \dfrac{\ell : (C \sqcup D)}{\ell : C \mid \ell : D}$

$(\exists)\ \dfrac{\ell : \exists R.C}{\ell : \exists R.\{\ell'\},\ \ \ell' : C}$ (ℓ' is new)

$(\neg\exists)\ \dfrac{\ell : \neg\exists R.C,\ \ \ell : \exists R.\{\ell'\}}{\ell' : \neg C}$

Rules for individuals

(sym) $\dfrac{\ell : \{\ell'\}}{\ell' : \{\ell\}}$ (¬sym) $\dfrac{\ell : \neg\{\ell'\}}{\ell' : \neg\{\ell\}}$ (ref) $\dfrac{\ell : C}{\ell : \{\ell\}}$

(mon) $\dfrac{\ell : \{\ell'\},\ \ \ell' : C}{\ell : C}$ (canc) $\dfrac{\ell : (\ell' : C)}{\ell' : C}$

$\ell : \{\ell'\} \equiv \ell = \ell'$

MANCHESTER 1824 The University of Manchester

Unrestricted Blocking Rule

$$\text{(ub)}\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Strategy conditions:

1. any rule is applied at most once to the same set of premises.
2. the (∃) rule is not applied to role assertion expressions.
3. if $\ell : \{\ell'\}$ in current branch and $\ell < \ell'$ then no applications of the (∃) rule to expressions $\ell' : \exists R.C$ are performed[1]
4. in every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of the (∃) rule

MANCHESTER 1824 The University of Manchester

[1] $<$ reflects the order in which the individuals are introduced

Unrestricted Blocking Rule

$$\text{(ub)}\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Strategy conditions:

1. any rule is applied at most once to the same set of premises.
2. the ($\exists$) rule is not applied to role assertion expressions.
3. if $\ell : \{\ell'\}$ in current branch and $\ell < \ell'$ then no applications of the ($\exists$) rule to expressions $\ell' : \exists R.C$ are performed[1]
4. in every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of the ($\exists$) rule

MANCHESTER 1824 The University of Manchester

[1] $<$ reflects the order in which the individuals are introduced

Unrestricted Blocking Rule

$$\text{(ub)}\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Strategy conditions:

1. any rule is applied at most once to the same set of premises.
2. the (∃) rule is not applied to role assertion expressions.
3. if $\ell : \{\ell'\}$ in current branch and $\ell < \ell'$ then no applications of the (∃) rule to expressions $\ell' : \exists R.C$ are performed[1]
4. in every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of the (∃) rule

[1] $<$ reflects the order in which the individuals are introduced

Unrestricted Blocking Rule

$$\text{(ub)}\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Strategy conditions:

1. any rule is applied at most once to the same set of premises.
2. the ($\exists$) rule is not applied to role assertion expressions.
3. if $\ell : \{\ell'\}$ in current branch and $\ell < \ell'$ then no applications of the ($\exists$) rule to expressions $\ell' : \exists R.C$ are performed[1]
4. in every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of the ($\exists$) rule

MANCHESTER 1824 The University of Manchester

[1] $<$ reflects the order in which the individuals are introduced

Unrestricted Blocking Rule

$$\text{(ub)}\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Strategy conditions:

1. any rule is applied at most once to the same set of premises.
2. the ($\exists$) rule is not applied to role assertion expressions.
3. if $\ell : \{\ell'\}$ in current branch and $\ell < \ell'$ then no applications of the ($\exists$) rule to expressions $\ell' : \exists R.C$ are performed[1]
4. in every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of the ($\exists$) rule

[1] $<$ reflects the order in which the individuals are introduced

Constructive Completeness and sub-Compatibility

Let $\mathcal{B}$ be an open branch in a tableau.

- $\ell \sim_{\mathcal{B}} \ell' \overset{\text{def}}{\Longleftrightarrow} \ell : \{\ell'\} \in \mathcal{B}$,
- $\Delta^{\mathcal{I}(\mathcal{B})} \overset{\text{def}}{=} \{\|\ell\| \mid \ell : \{\ell\} \in \mathcal{B}\}$.

A tableau calculus T_L is *constructively complete* for L iff for any satisfiable concept C and any open branch $\mathcal{B}$ in $T_L(C)$ there is an L-model $\mathcal{I}(\mathcal{B}) = (\Delta^{\mathcal{I}(\mathcal{B})}, \cdot^{\mathcal{I}(\mathcal{B})})$ such that

- $\ell : D \in \mathcal{B}$ implies $\|\ell\| \in D^{\mathcal{I}(\mathcal{B})}$, and
- $\ell : \exists R.\{\ell'\} \in \mathcal{B}$ implies $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}(\mathcal{B})}$.

T_L is *compatible with* sub iff for any concept C and $\ell : D$ in $T_L(C)$ either

- $D \in \text{sub}(C)$, or
- $D = \{\ell'\}$, or $D = \neg\{\ell'\}$, or
- $D = \exists R.\{\ell'\}$, or $D = \neg\exists R.\{\ell'\}$, for some role $R \in \text{sub}(C)$.

MANCHESTER 1824 The University of Manchester

Constructive Completeness and sub-Compatibility

Let $\mathcal{B}$ be an open branch in a tableau.

- $\ell \sim_{\mathcal{B}} \ell' \overset{\text{def}}{\Longleftrightarrow} \ell : \{\ell'\} \in \mathcal{B}$,
- $\Delta^{\mathcal{I}(\mathcal{B})} \overset{\text{def}}{=} \{\|\ell\| \mid \ell : \{\ell\} \in \mathcal{B}\}$.

A tableau calculus T_L is *constructively complete* for L iff for any satisfiable concept C and any open branch $\mathcal{B}$ in $T_L(C)$ there is an L-model $\mathcal{I}(\mathcal{B}) = (\Delta^{\mathcal{I}(\mathcal{B})}, .^{\mathcal{I}(\mathcal{B})})$ such that

- $\ell : D \in \mathcal{B}$ implies $\|\ell\| \in D^{\mathcal{I}(\mathcal{B})}$, and
- $\ell : \exists R.\{\ell'\} \in \mathcal{B}$ implies $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}(\mathcal{B})}$.

T_L is *compatible with* sub iff for any concept C and $\ell : D$ in $T_L(C)$ either

- $D \in \text{sub}(C)$, or
- $D = \{\ell'\}$, or $D = \neg\{\ell'\}$, or
- $D = \exists R.\{\ell'\}$, or $D = \neg\exists R.\{\ell'\}$, for some role $R \in \text{sub}(C)$.

Constructive Completeness and sub-Compatibility

Let $\mathcal{B}$ be an open branch in a tableau.

- $\ell \sim_{\mathcal{B}} \ell' \stackrel{\text{def}}{\Longleftrightarrow} \ell : \{\ell'\} \in \mathcal{B}$,
- $\Delta^{\mathcal{I}(\mathcal{B})} \stackrel{\text{def}}{=} \{\|\ell\| \mid \ell : \{\ell\} \in \mathcal{B}\}$.

A tableau calculus T_L is *constructively complete* for L iff for any satisfiable concept C and any open branch $\mathcal{B}$ in $T_L(C)$ there is an L-model $\mathcal{I}(\mathcal{B}) = (\Delta^{\mathcal{I}(\mathcal{B})}, \cdot^{\mathcal{I}(\mathcal{B})})$ such that

- $\ell : D \in \mathcal{B}$ implies $\|\ell\| \in D^{\mathcal{I}(\mathcal{B})}$, and
- $\ell : \exists R.\{\ell'\} \in \mathcal{B}$ implies $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}(\mathcal{B})}$.

T_L is *compatible with* sub iff for any concept C and $\ell : D$ in $T_L(C)$ either

- $D \in \text{sub}(C)$, or
- $D = \{\ell'\}$, or $D = \neg\{\ell'\}$, or
- $D = \exists R.\{\ell'\}$, or $D = \neg\exists R.\{\ell'\}$, for some role $R \in \text{sub}(C)$.

The Main Theorem

Theorem

Let L be a (description) logic. $T_L +$ (ub) is sound, complete, and terminating tableau calculus for L, if the following conditions all hold:

1. sub *is a finite closure operator for L-expressions.*
2. *L is a logic which admits finite filtration.*
3. *T_L is a sound and constructively complete tableau calculus for L and is compatible with* sub.

Sound and Constructively Complete Tableau Calculus for $\mathcal{SO}$

$T_{\mathcal{SO}}$ contains the common tableau rules and the following rules for every $s \in$ Trans:

$$(\mathsf{Trans}_s)\ \frac{\ell : \exists s.\{\ell'\},\quad \ell' : \exists s.\{\ell''\}}{\ell : \exists s.\{\ell''\}}$$

- Soundness is trivial.
- Constructive completeness is easy.
- Clearly, $T_{\mathcal{SO}}$ is compatible with the subexpression operator sub.

Theorem

$T_{\mathcal{SO}}$ + (ub) *is sound, complete, and terminating.*

Sound and Constructively Complete Tableau Calculus for $\mathcal{SO}$

$T_{\mathcal{SO}}$ contains the common tableau rules and the following rules for every $s \in$ Trans:

$$(\text{Trans}_s)\ \frac{\ell : \exists s.\{\ell'\}, \quad \ell' : \exists s.\{\ell''\}}{\ell : \exists s.\{\ell''\}}$$

- Soundness is trivial.
- Constructive completeness is easy.
- Clearly, $T_{\mathcal{SO}}$ is compatible with the subexpression operator sub.

Theorem

$T_{\mathcal{SO}} +$ (ub) *is sound, complete, and terminating.*

MANCHESTER 1824 The University of Manchester

Sound and Constructively Complete Tableau Calculus for $\mathcal{SO}$

$T_{\mathcal{SO}}$ contains the common tableau rules and the following rules for every $s \in$ Trans:

$$(\mathsf{Trans}_s)\ \frac{\ell : \exists s.\{\ell'\},\quad \ell' : \exists s.\{\ell''\}}{\ell : \exists s.\{\ell''\}}$$

- Soundness is trivial.
- Constructive completeness is easy.
- Clearly, $T_{\mathcal{SO}}$ is compatible with the subexpression operator sub.

Theorem

$T_{\mathcal{SO}} +$ (ub) *is sound, complete, and terminating.*

Sound and Constructively Complete Tableau Calculi for $\mathcal{ALBO}$

$T_{\mathcal{ALBO}}$ contains the common tableau rules and the following rules for complex role operators:

Positive Role Occurrences

$$(\exists\sqcup)\ \frac{\ell : \exists(R \sqcup S).\{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell : \exists S.\{\ell'\}}$$

$$(\exists^{-1})\ \frac{\ell : \exists R^{-1}.\{\ell'\}}{\ell' : \exists R.\{\ell\}}$$

$$(\exists\neg)\ \frac{\ell : \exists\neg R.\{\ell'\}}{\ell : \neg\exists R.\{\ell'\}}$$

Negative Role Occurrences

$$(\neg\exists\sqcup)\ \frac{\ell : \neg\exists(R \sqcup S).C}{\ell : \neg\exists R.C,\ \ell : \neg\exists S.C}$$

$$(\neg\exists^{-1})\ \frac{\ell : \neg\exists R^{-1}.C,\ \ell' : \exists R.\{\ell\}}{\ell' : \neg C}$$

$$(\neg\exists\neg)\ \frac{\ell : \neg\exists\neg R.C,\ \ell' : \{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell' : \neg C}$$

$T^q_{\mathcal{ALBO}} \stackrel{\text{def}}{=} T_{\mathcal{ALBO}} - (\exists\neg)$

- Both calculi are sound and compatible with the subexpression operator sub.
- $T_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-models.
- $T^q_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-quasi-models.

Theorem

- $T_{\mathcal{ALBO}} +$ (ub) *is sound, complete w.r.t. $\mathcal{ALBO}$-models, and terminating.*
- $T^q_{\mathcal{ALBO}} +$ (ub) *is sound, complete w.r.t. $\mathcal{ALBO}$-quasi-models, and terminating.*

Sound and Constructively Complete Tableau Calculi for $\mathcal{ALBO}$

$T_{\mathcal{ALBO}}$ contains the common tableau rules and the following rules for complex role operators:

Positive Role Occurrences

$$(\exists\sqcup)\ \frac{\ell : \exists(R \sqcup S).\{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell : \exists S.\{\ell'\}}$$

$$(\exists^{-1})\ \frac{\ell : \exists R^{-1}.\{\ell'\}}{\ell' : \exists R.\{\ell\}}$$

$$(\exists\neg)\ \frac{\ell : \exists\neg R.\{\ell'\}}{\ell : \neg\exists R.\{\ell'\}}$$

Negative Role Occurrences

$$(\neg\exists\sqcup)\ \frac{\ell : \neg\exists(R \sqcup S).C}{\ell : \neg\exists R.C,\ \ \ell : \neg\exists S.C}$$

$$(\neg\exists^{-1})\ \frac{\ell : \neg\exists R^{-1}.C,\ \ \ell' : \exists R.\{\ell\}}{\ell' : \neg C}$$

$$(\neg\exists\neg)\ \frac{\ell : \neg\exists\neg R.C,\ \ \ell' : \{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell' : \neg C}$$

$T^q_{\mathcal{ALBO}} \stackrel{\text{def}}{=} T_{\mathcal{ALBO}} - (\exists\neg)$

- Both calculi are sound and compatible with the subexpression operator sub.
- $T_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-models.
- $T^q_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-quasi-models.

Theorem

- *$T_{\mathcal{ALBO}}$ + (ub) is sound, complete w.r.t. $\mathcal{ALBO}$-models, and terminating.*
- *$T^q_{\mathcal{ALBO}}$ + (ub) is sound, complete w.r.t. $\mathcal{ALBO}$-quasi-models, and terminating.*

Sound and Constructively Complete Tableau Calculi for $\mathcal{ALBO}$

$T_{\mathcal{ALBO}}$ contains the common tableau rules and the following rules for complex role operators:

Positive Role Occurrences

$$(\exists\sqcup)\ \frac{\ell : \exists(R \sqcup S).\{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell : \exists S.\{\ell'\}}$$

$$(\exists^{-1})\ \frac{\ell : \exists R^{-1}.\{\ell'\}}{\ell' : \exists R.\{\ell\}}$$

$$(\exists\neg)\ \frac{\ell : \exists\neg R.\{\ell'\}}{\ell : \neg\exists R.\{\ell'\}}$$

Negative Role Occurrences

$$(\neg\exists\sqcup)\ \frac{\ell : \neg\exists(R \sqcup S).C}{\ell : \neg\exists R.C,\ \ \ell : \neg\exists S.C}$$

$$(\neg\exists^{-1})\ \frac{\ell : \neg\exists R^{-1}.C,\ \ \ell' : \exists R.\{\ell\}}{\ell' : \neg C}$$

$$(\neg\exists\neg)\ \frac{\ell : \neg\exists\neg R.C,\ \ \ell' : \{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell' : \neg C}$$

$T^q_{\mathcal{ALBO}} \stackrel{\text{def}}{=} T_{\mathcal{ALBO}} - (\exists\neg)$

- Both calculi are sound and compatible with the subexpression operator sub.
- $T_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-models.
- $T^q_{\mathcal{ALBO}}$ is constructively complete w.r.t. $\mathcal{ALBO}$-quasi-models.

Theorem

- $T_{\mathcal{ALBO}} +$ (ub) *is sound, complete w.r.t. $\mathcal{ALBO}$-models, and terminating.*
- $T^q_{\mathcal{ALBO}} +$ (ub) *is sound, complete w.r.t. $\mathcal{ALBO}$-quasi-models, and terminating.*

Outline

MANCHESTER 1824 The University of Manchester

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

MANCHESTER 1824 The University of Manchester

Conclusion

- A general method for turning ground semantic tableau calculi into decision procedures is introduced.
- The method is illustrated on two examples: $\mathcal{SO}$ and $\mathcal{ALBO}$.
- The method is not limited by description logic language.
- It works for other ground tableau and similar decision approaches.
- The framework provides a basis for enhancing prover engineering platforms with a flexible blocking mechanism with which more general tableau decision procedures can be constructed.
- The approach also provides the theoretical background for the way blocking is implemented in the METTEL system.
- The framework is a first step towards the ambitious goal of automated generation of provers for decidable logics.

k you! Questions?

nk you! Questions?

MANCHESTER 1824 The University of Manchester

ank you! Questions?

ank you! Questions?

Thank you! Questions?

Thank you! Questions?

Thank you! Questions?

Thank you! Questions?

MANCHESTER 1824
The University of Manchester

Thank you! Questions?

Thank you! Questions?

Thank you! Questions?

Thank you! Questions?

Thank you! Questions?

The University of Manchester MANCHESTER 1824

Thank you! Questions?