Inference in Description Logics

Petr Křemen
petr.kremen@fel.cvut.cz

FEL ČVUT

Our plan

Inference Problems

Inference Algorithms
    Tableau Algorithm for $\mathcal{ALC}$

Inference Problems

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts C, D, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C *subsumes* concept D, i.e. $\mathcal{T} \models D \sqsubseteq C$?

(equivalence) two concepts C and D are *equivalent*, i.e. $\mathcal{T} \models C \equiv D$?

(disjoint) two concepts C and D are *disjoint*, i.e. $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept …

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts C, D, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C *subsumes* concept D, i.e. $\mathcal{T} \models D \sqsubseteq C$?

(equivalence) two concepts C and D are *equivalent*, i.e. $\mathcal{T} \models C \equiv D$?

(disjoint) two concepts C and D are *disjoint*, i.e. $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept ...

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts C, D, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C *subsumes* concept D, i.e. $\mathcal{T} \models D \sqsubseteq C$?

(equivalence) two concepts C and D are *equivalent*, i.e. $\mathcal{T} \models C \equiv D$?

(disjoint) two concepts C and D are *disjoint*, i.e. $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept ...

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts C, D, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C *subsumes* concept D, i.e. $\mathcal{T} \models D \sqsubseteq C$?

(equivalence) two concepts C and D are *equivalent*, i.e. $\mathcal{T} \models C \equiv D$?

(disjoint) two concepts C and D are *disjoint*, i.e. $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept ...

Inference Problems in TBOX

We have introduced syntax and semantics of the language $\mathcal{ALC}$. Now, let's look on automated reasoning. Having a $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For TBOX $\mathcal{T}$ and concepts C, D, we want to decide whether

(unsatisfiability) concept C is *unsatisfiable*, i.e. $\mathcal{T} \models C \sqsubseteq \bot$?

(subsumption) concept C *subsumes* concept D, i.e. $\mathcal{T} \models D \sqsubseteq C$?

(equivalence) two concepts C and D are *equivalent*, i.e. $\mathcal{T} \models C \equiv D$?

(disjoint) two concepts C and D are *disjoint*, i.e. $\mathcal{T} \models C \sqcap D \sqsubseteq \bot$?

All these tasks can be reduced to unsatisfiability checking of a single concept ...

Reduction to Concept Unsatisfiability – Example

Example

These reductions are straighforward – let's show, how to reduce subsumption checking to unsatisfiability checking. Reduction of other inference problems to unsatisfiability is analogous.

$$
\begin{array}{rccl}
(\mathcal{T} \models C \sqsubseteq D) & & & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \text{implies} & \mathcal{I} \models C \sqsubseteq D) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \text{implies} & C^{\mathcal{I}} \subseteq D^{\mathcal{I}}) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \text{implies} & C^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) \subseteq \emptyset & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} & \text{implies} & \mathcal{I} \models C \sqcap \neg D \sqsubseteq \bot & \text{iff} \\
(\mathcal{T} \models C \sqcap \neg D \sqsubseteq \bot) & & &
\end{array}
$$

Inference Problems for ABOX

... for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

… for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

… for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

... for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

... for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

... for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Problems for ABOX

... for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals a,a_0 we want to decide whether

(consistency checking) ABOX $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ (in short if $\mathcal{K}$ is consistent).

(instance checking) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

(role checking) $\mathcal{T} \cup \mathcal{A} \models R(a, a_0)$?

(instance retrieval) find all individuals a_1, for which $\mathcal{T} \cup \mathcal{A} \models C(a_1)$.

realization find the most specific concept C from a set of concepts, such that $\mathcal{T} \cup \mathcal{A} \models C(a)$.

All these tasks, as well as concept unsatisfiability checking, can be reduced to consistency checking. Under which condition and how ?

Inference Algorithms

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [BCM+03].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite, see [HS03], [HS01], [BCM+03].

other ... – e.g. resolution-based [Hab06], transformation to finite automata [BCM+03], etc.

We will introduce tableau algorithms.

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [BCM^{+}03].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite, see [HS03], [HS01], [BCM^{+}03].

other ... – e.g. resolution-based [Hab06], transformation to finite automata [BCM^{+}03], etc.

We will introduce tableau algorithms.

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [BCM$^+$03].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite, see [HS03], [HS01], [BCM$^+$03].

other ... – e.g. resolution-based [Hab06], transformation to finite automata [BCM$^+$03], etc.

We will introduce tableau algorithms.

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [BCM+03].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite, see [HS03], [HS01], [BCM+03].

other ... – e.g. resolution-based [Hab06], transformation to finite automata [BCM+03], etc.

We will introduce tableau algorithms.

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.”**
- Each TA can be seen as a *production system* :
  - state of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - inference rules ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists$, $\sqcap$, etc. They serve to modify the completion graphs according to
  - chosen strategy for rule application

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.”**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists$, $\sqcap$, etc. and serve to modify the completion graphs according to
  - choosen strategy for rule application

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.”**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists, \sqcap$, etc. and serve to modify the completion graphs according to
  - choosen *strategy* for rule application

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$."**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists, \sqcap$, etc. and serve to modify the completion graphs according to
  - choosen *strategy* for rule application

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.”**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists, \sqcap$, etc. and serve to modify the completion graphs according to
  - choosen *strategy* for rule application

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking ABOXu consistency checking w.r.t. an TBOXu. TAs are not new in DL – they were known for FOL as well.
- Main idea is simple: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$."**
- Each TA can be seen as a *production system* :
  - *state* of TA ($\sim$ data base) is made up by a set of completion graphs (see next slide),
  - *inference rules* ($\sim$ production rules) implement semantics of particular constructs of the given language, e.g. $\exists, \sqcap$, etc. and serve to modify the completion graphs according to
  - choosen *strategy* for rule application

Completion Graphs

completion graph is a labeled oriented graph $G = (V_G, E_G, L_G))$, where each node $x \in V_G$ is labeled with a set $L_G(x)$ of concepts and each edge $\langle x, y \rangle \in E_G$ is labeled with a set of edges $L_G(\langle x, y \rangle)$[5]

direct clash occurs in a completion graph $G = (V_G, E_G, L_G))$, if $\{A, \neg A\} \subseteq L_G(x)$, or $\bot \in L_G(x)$, for some atomic concept A and a node $x \in V_G$

complete completion graph is a completion graph $G = (V_G, E_G, L_G))$, to which no completion rule from the set of TA completion rules can be applied.

Do not mix with notion of complete graphs known from graph theory.

---

[5] Next in the text the notation is often shortened as $L_G(x, y)$ instead of $L_G(\langle x, y \rangle)$.

Completion Graphs

completion graph is a labeled oriented graph $G = (V_G, E_G, L_G))$, where each node $x \in V_G$ is labeled with a set $L_G(x)$ of concepts and each edge $\langle x, y\rangle \in E_G$ is labeled with a set of edges $L_G(\langle x, y\rangle)$[5]

direct clash occurs in a completion graph $G = (V_G, E_G, L_G))$, if $\{A, \neg A\} \subseteq L_G(x)$, or $\perp \in L_G(x)$, for some atomic concept A and a node $x \in V_G$

complete completion graph is a completion graph $G = (V_G, E_G, L_G))$, to which no completion rule from the set of TA completion rules can be applied.

Do not mix with notion of complete graphs known from graph theory.

---

[5] Next in the text the notation is often shortened as $L_G(x, y)$ instead of $L_G(\langle x, y\rangle)$.

Completion Graphs

completion graph is a labeled oriented graph $G = (V_G, E_G, L_G))$, where each node $x \in V_G$ is labeled with a set $L_G(x)$ of concepts and each edge $\langle x, y\rangle \in E_G$ is labeled with a set of edges $L_G(\langle x, y\rangle)$[5]

direct clash occurs in a completion graph $G = (V_G, E_G, L_G))$, if $\{A, \neg A\} \subseteq L_G(x)$, or $\perp \in L_G(x)$, for some atomic concept A and a node $x \in V_G$

complete completion graph is a completion graph $G = (V_G, E_G, L_G))$, to which no completion rule from the set of TA completion rules can be applied.

Do not mix with notion of complete graphs known from graph theory.

---

[5] Next in the text the notation is often shortened as $L_G(x, y)$ instead of $L_G(\langle x, y\rangle)$.

Completion Graphs

completion graph is a labeled oriented graph $G = (V_G, E_G, L_G))$, where each node $x \in V_G$ is labeled with a set $L_G(x)$ of concepts and each edge $\langle x, y \rangle \in E_G$ is labeled with a set of edges $L_G(\langle x, y \rangle)$[5]

direct clash occurs in a completion graph $G = (V_G, E_G, L_G))$, if $\{A, \neg A\} \subseteq L_G(x)$, or $\bot \in L_G(x)$, for some atomic concept A and a node $x \in V_G$

complete completion graph is a completion graph $G = (V_G, E_G, L_G))$, to which no completion rule from the set of TA completion rules can be applied.

Do not mix with notion of complete graphs known from graph theory.

---

[5] Next in the text the notation is often shortened as $L_G(x, y)$ instead of $L_G(\langle x, y \rangle)$.

Completion Graphs (2)

We define also $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, where $\mathcal{A}_G$ is an ABOX constructed from G, as follows

- $C(a)$ for each node $a \in V_G$ and each concept $C \in L_G(a)$ and
- $R(a, b)$ for each edge $\langle a, b \rangle \in E_G$ and each role $R \in L_G(a, b)$ and

Completion Graphs (2)

We define also $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, where $\mathcal{A}_G$ is an ABOX constructed from G, as follows

- $C(a)$ for each node $a \in V_G$ and each concept $C \in L_G(a)$ and
- $R(a, b)$ for each edge $\langle a, b \rangle \in E_G$ and each role $R \in L_G(a, b)$ and

Completion Graphs (2)

We define also $\mathcal{I} \models G$ iff $\mathcal{I} \models \mathcal{A}_G$, where $\mathcal{A}_G$ is an ABOX constructed from G, as follows

- $C(a)$ for each node $a \in V_G$ and each concept $C \in L_G(a)$ and
- $R(a, b)$ for each edge $\langle a, b \rangle \in E_G$ and each role $R \in L_G(a, b)$ and

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B$).

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:

- for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
- for each $R(a, b)$ put $(a, b) \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
- Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B)$.

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:
   - for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
   - for each $R(a, b)$ put $(a, b) \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
   - Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B$).

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:

- for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
- for each $R(a, b)$ put $\langle a, b \rangle \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
- Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B$).

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:
   - for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
   - for each $R(a, b)$ put $\langle a, b\rangle \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
   - Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0 (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B)$.

1 (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:
   - for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
   - for each $R(a, b)$ put $\langle a, b \rangle \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
   - Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau Algorithm for $\mathcal{ALC}$ with empty TBOX

let's have $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For a moment, consider for simplicity that $\mathcal{T} = \emptyset$.

0. (Preprocessing) Transform all concepts appearing in $\mathcal{K}$ to the "negational normal form" (NNF) by equivalent operations known from propositional and predicate logics. As a result, all concepts contain negation $\neg$ at most just before atomic concepts, e.g. $\neg(A \sqcap B)$ is equivalent (de Morgan rules) as $\neg A \sqcup \neg B$).
1. (Initialization) Initial state of the algorithm is $S_0 = \{G_0\}$, where $G_0 = (V_{G_0}, E_{G_0}, L_{G_0})$ is made up from $\mathcal{A}$ as follows:
    - for each $C(a)$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
    - for each $R(a, b)$ put $\langle a, b\rangle \in E_{G_0}$ and $R \in L_{G_0}(a, b)$
    - Sets $V_{G_0}, E_{G_0}, L_{G_0}$ are smallest possible with these properties.

Tableau algorithm for $\mathcal{ALC}$ without TBOX (2)

...

2. (Consistency Check) Current algorithm state is S. If each $G \in S$ contains a direct clash, terminate with result "INCONSISTENT"
3. (Model Check) Let's choose one $G \in S$ that doesn't contain a direct clash. If G is complete w.r.t. rules shown next, the algorithm terminates with result "CONSISTENT"
4. (Rule Application) Find a rule that is applicable to G and apply it. As a result, we obtain from the state S a new state S'. Jump to step 2.

Tableau algorithm for $\mathcal{ALC}$ without TBOX (2)

...

2. (Consistency Check) Current algorithm state is S. If each $G \in S$ contains a direct clash, terminate with result "INCONSISTENT"
3. (Model Check) Let's choose one $G \in S$ that doesn't contain a direct clash. If G is complete w.r.t. rules shown next, the algorithm terminates with result "CONSISTENT"
4. (Rule Application) Find a rule that is applicable to G and apply it. As a result, we obtain from the state S a new state S'. Jump to step 2.

Tableau algorithm for $\mathcal{ALC}$ without TBOX (2)

...

2. (Consistency Check) Current algorithm state is S. If each $G \in S$ contains a direct clash, terminate with result "INCONSISTENT"
3. (Model Check) Let's choose one $G \in S$ that doesn't contain a direct clash. If G is complete w.r.t. rules shown next, the algorithm terminates with result "CONSISTENT"
4. (Rule Application) Find a rule that is applicable to G and apply it. As a result, we obtain from the state S a new state S'. Jump to step 2.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{1|2} = (V_G, E_G, L_{G_{1|2}})$, and $L_{G_{1|2}}(a) = L_G(a) \cup \{C_{1|2}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, and $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$
then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{1|2} = (V_G, E_G, L_{G_{1|2}})$, and
$L_{G_{1|2}}(a) = L_G(a) \cup \{C_{1|2}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and
at the same time $C \in L_G(b)$,
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{(a, b)\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at
the same time $C \notin L_G(b)$,
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.
then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{1(2)} = (V_G, E_G, L_{G_{1(2)}})$, and
$L_{G_{1(2)}}(a) = L_G(a) \cup \{C_{1(2)}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and
at the same time $C \in L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at
the same time $C \notin L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{(a, b)\}, L_{G'})$, and $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{(a, b)\}, L_{G'})$, and $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{D\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.
then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and
$L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and
at the same time $C \in L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at
the same time $C \notin L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and
$L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and
at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at
the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{C\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and
$L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{D\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \not\subseteq L_G(a)$ for some $a \in V_G$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.
then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and
$L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a
$L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and
$L_{G'}(b) = L_G(b) \cup \{D\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_{\sqcup}$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_{\exists}$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{D\}$ and otherwise is the same as L_G.

TA for $\mathcal{ALC}$ without TBOX – Inference Rules

$\rightarrow_\sqcap$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a) = L_G(a) \cup \{C_1, C_2\}$ and otherwise is the same as L_G.

$\rightarrow_\sqcup$ rule

if $(C_1 \sqcup C_2) \in L_G(a)$ and $\{C_1, C_2\} \cap L_G(a) = \emptyset$ for some $a \in V_G$.

then $S' = S \cup \{G_1, G_2\} \setminus \{G\}$, where $G_{(1|2)} = (V_G, E_G, L_{G_{(1|2)}})$, and $L_{G_{(1|2)}}(a) = L_G(a) \cup \{C_{(1|2)}\}$ and otherwise is the same as L_G.

$\rightarrow_\exists$ rule

if $(\exists R \cdot C) \in L_G(a)$ and there exists no $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \in L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{b\}, E_G \cup \{\langle a, b\rangle\}, L_{G'})$, a $L_{G'}(b) = \{C\}$, $L_{G'}(a, b) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a)$ and there exists $b \in V_G$ such that $R \in L_G(a, b)$ and at the same time $C \notin L_G(b)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(b) = L_G(b) \cup \{D\}$ and otherwise is the same as L_G.

Finiteness

Finiteness of the TA is an easy consequence of the following:

- $\mathcal{K}$ is finite
- in each step, TA state can be enriched at most by one completion graph (only by application of $\rightarrow_{\sqcup}$ rule). Number of disjunctions ($\sqcup$) in $\mathcal{K}$ is finite, i.e. the $\sqcup$ can be applied just finite number of times.
- for each completion graph $G = (V_G, E_G, L_G)$ it holds that number of nodes in V_G is less or equal to the number of individuals in $\mathcal{A}$ plus number of existential quantifiers in $\mathcal{A}$.
- after application of any of the following rules $\rightarrow_{\sqcap}, \rightarrow_{\exists}, \rightarrow_{\forall}$ graph G is either enriched with a new node, new edge, or labeling of an existing node/edge is enriched. All these operations are finite.

Finiteness

Finiteness of the TA is an easy consequence of the following:

- $\mathcal{K}$ is finite
- in each step, TA state can be enriched at most by one completion graph (only by application of $\rightarrow_{\sqcup}$ rule). Number of disjunctions ($\sqcup$) in $\mathcal{K}$ is finite, i.e. the $\sqcup$ can be applied just finite number of times.
- for each completion graph $G = (V_G, E_G, L_G)$ it holds that number of nodes in V_G is less or equal to the number of individuals in $\mathcal{A}$ plus number of existential quantifiers in $\mathcal{A}$.
- after application of any of the following rules $\rightarrow_{\sqcap}, \rightarrow_{\exists}, \rightarrow_{\forall}$ graph G is either enriched with a new node, new edge, or labeling of an existing node/edge is enriched. All these operations are finite.

Finiteness

Finiteness of the TA is an easy consequence of the following:

- $\mathcal{K}$ is finite
- in each step, TA state can be enriched at most by one completion graph (only by application of $\rightarrow_{\sqcup}$ rule). Number of disjunctions ($\sqcup$) in $\mathcal{K}$ is finite, i.e. the $\sqcup$ can be applied just finite number of times.
- for each completion graph $G = (V_G, E_G, L_G)$ it holds that number of nodes in V_G is less or equal to the number of individuals in $\mathcal{A}$ plus number of existential quantifiers in $\mathcal{A}$.
- after application of any of the following rules $\rightarrow_{\sqcap}, \rightarrow_{\exists}, \rightarrow_{\forall}$ graph G is either enriched with a new node, new edge, or labeling of an existing node/edge is enriched. All these operations are finite.

Finiteness

Finiteness of the TA is an easy consequence of the following:

- $\mathcal{K}$ is finite
- in each step, TA state can be enriched at most by one completion graph (only by application of $\rightarrow_{\sqcup}$ rule). Number of disjunctions ($\sqcup$) in $\mathcal{K}$ is finite, i.e. the $\sqcup$ can be applied just finite number of times.
- for each completion graph $G = (V_G, E_G, L_G)$ it holds that number of nodes in V_G is less or equal to the number of individuals in $\mathcal{A}$ plus number of existential quantifiers in $\mathcal{A}$.
- after application of any of the following rules $\rightarrow_{\sqcap}, \rightarrow_{\exists}, \rightarrow_{\forall}$ graph G is either enriched with a new node, new edge, or labeling of an existing node/edge is enriched. All these operations are finite.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
  - As a result $a^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
  - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a^{\mathcal{I}}, i \rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
  - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b \rangle$ and node b has been adjusted.
  - It is enough to place $i = b^{\mathcal{I}}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
  - As a result $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a^\mathcal{I}, i\rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.
  - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b\rangle$ and node b has been adjusted.
  - It is enough to place $i = b^\mathcal{I}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_{\exists}$ rule:
    - Before application of $\rightarrow_{\exists}$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
    - As a result $a^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
    - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a^{\mathcal{I}}, i\rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
    - By application of $\rightarrow_{\exists}$ a new node b was created in G_{i+1} and the label of edge $\langle a, b\rangle$ and node b has been adjusted.
    - It is enough to place $i = b^{\mathcal{I}}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
  - As a result $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.
  - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b \rangle$ and node b has been adjusted.
  - It is enough to place $i = b^\mathcal{I}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
  - As a result $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.
  - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b \rangle$ and node b has been adjusted.
  - It is enough to place $i = b^\mathcal{I}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
  - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
  - As a result $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.
  - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b \rangle$ and node b has been adjusted.
  - It is enough to place $i = b^\mathcal{I}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Soundness

- Soundness of the TA can be verified as follows. For any $\mathcal{I} \models \mathcal{A}_{G_i}$, it must hold that $\mathcal{I} \models \mathcal{A}_{G_{i+1}}$. We have to show that application of each rule preserves consistency. As an example, let's take the $\rightarrow_\exists$ rule:
    - Before application of $\rightarrow_\exists$ rule, $(\exists R \cdot C) \in L_{G_i}(a)$ held for $a \in V_{G_i}$.
    - As a result $a^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
    - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a^\mathcal{I}, i \rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.
    - By application of $\rightarrow_\exists$ a new node b was created in G_{i+1} and the label of edge $\langle a, b \rangle$ and node b has been adjusted.
    - It is enough to place $i = b^\mathcal{I}$ to see that after rule application the domain element (necessary present in any interpretation because of $\exists$ construct semantics) has been "materialized". As a result, the rule is correct.
- For other rules, the soundness is shown in a similar way.

Completeness

- To prove completeness of the TA, it is necessary to construct a model for each complete completion graph G that doesn't contain a direct clash. Canonical model $\mathcal{I}$ can be constructed as follows:
  - the domain $\Delta^{\mathcal{I}}$ will consist of all nodes of G.
  - for each atomic concept A let's define $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
  - for each atomic role R let's define $R^{\mathcal{I}} = \{\langle a, b\rangle \mid R \in L_G(a, b)\}$
- Observe that $\mathcal{I}$ is a model of $\mathcal{A}_G$. A backward induction can be used to show that $\mathcal{I}$ must be also a model of each previous step and thus also $\mathcal{A}$.

Completeness

- To prove completeness of the TA, it is necessary to construct a model for each complete completion graph G that doesn't contain a direct clash. Canonical model $\mathcal{I}$ can be constructed as follows:
    - the domain $\Delta^{\mathcal{I}}$ will consist of all nodes of G.
    - for each atomic concept A let's define $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
    - for each atomic role R let's define $R^{\mathcal{I}} = \{\langle a, b\rangle \mid R \in L_G(a, b)\}$
- Observe that $\mathcal{I}$ is a model of $\mathcal{A}_G$. A backward induction can be used to show that $\mathcal{I}$ must be also a model of each previous step and thus also $\mathcal{A}$.

Completeness

- To prove completeness of the TA, it is necessary to construct a model for each complete completion graph G that doesn't contain a direct clash. Canonical model $\mathcal{I}$ can be constructed as follows:
  - the domain $\Delta^{\mathcal{I}}$ will consist of all nodes of G.
  - for each atomic concept A let's define $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
  - for each atomic role R let's define $R^{\mathcal{I}} = \{\langle a, b\rangle \mid R \in L_G(a, b)\}$
- Observe that $\mathcal{I}$ is a model of $\mathcal{A}_G$. A backward induction can be used to show that $\mathcal{I}$ must be also a model of each previous step and thus also $\mathcal{A}$.

A few remarks on TAs

- Why we need completion graphs ? Aren't ABOXes enough to maintain the state for TA ?
  - indeed, for $\mathcal{ALC}$ they would be enough. However, for complex DLs a TA state cannot be stored in an ABOX.
- What about complexity of the algorithm ?
  - Without proof, let's state that the algorithm is in P-SPACE (between NP and EXP-TIME).

A few remarks on TAs

- Why we need completion graphs ? Aren't ABOXes enough to maintain the state for TA ?
  - indeed, for $\mathcal{ALC}$ they would be enough. However, for complex DLs a TA state cannot be stored in an ABOX.
- What about complexity of the algorithm ?
  - Without proof, let's state that the algorithm is in P-SPACE (between NP and EXP-TIME).

A few remarks on TAs

- Why we need completion graphs ? Aren't ABOXes enough to maintain the state for TA ?
    - indeed, for $\mathcal{ALC}$ they would be enough. However, for complex DLs a TA state cannot be stored in an ABOX.
- What about complexity of the algorithm ?
    - Without proof, let's state that the algorithm is in P-SPACE (between NP and EXP-TIME).

A few remarks on TAs

- Why we need completion graphs ? Aren't ABOXes enough to maintain the state for TA ?
  - indeed, for $\mathcal{ALC}$ they would be enough. However, for complex DLs a TA state cannot be stored in an ABOX.
- What about complexity of the algorithm ?
  - Without proof, let's state that the algorithm is in P-SPACE (between NP and EXP-TIME).

TA Run Example

Example

Let's check consistency of the ontology $\mathcal{K}_2 = (\emptyset, \mathcal{A}_2)$, where $\mathcal{A}_2 = \{(\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \neg\exists maDite \cdot (Muz \sqcap Prarodic))(JAN)\}$.

- Let's transform the concept into NNF: $\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \forall maDite \cdot (\neg Muz \sqcup \neg Prarodic)$
- Initial state G_0 of the TA is

"JAN"

$((\forall$ maDite - $(\neg$Muz $\sqcup$ $\neg$Prarodic$))$ $\sqcap$ $(\exists$ maDite - Prarodic$)$ $\sqcap$ $(\exists$ maDite - Muz$))$

TA Run Example

Example

Let's check consistency of the ontology $\mathcal{K}_2 = (\emptyset, \mathcal{A}_2)$, where $\mathcal{A}_2 = \{(\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \neg\exists maDite \cdot (Muz \sqcap Prarodic))(JAN)\}$.

- Let's transform the concept into NNF: $\exists maDite \cdot Muz \sqcap \exists maDite \cdot Prarodic \sqcap \forall maDite \cdot (\neg Muz \sqcup \neg Prarodic)$
- Initial state G_0 of the TA is

"JAN"

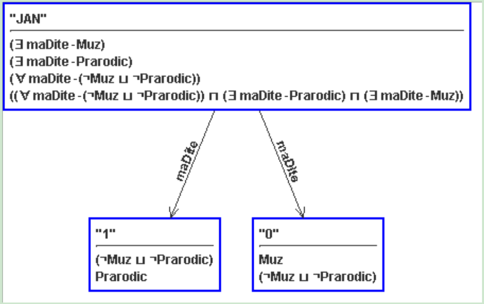
((∀ maDite -(¬Muz ⊔ ¬Prarodic)) ⊓ (∃ maDite -Prarodic) ⊓ (∃ maDite -Muz))

TA Run Example (2)

Example

. . .

- Now, four sequences of steps 2,3,4 of the TA are performed. TA state in step 4, evolves as follows:
- $\{G_0\} \xrightarrow{\sqcap\text{-rule}} \{G_1\} \xrightarrow{\exists\text{-rule}} \{G_2\} \xrightarrow{\exists\text{-rule}} \{G_3\} \xrightarrow{\forall\text{-rule}} \{G_4\}$, where G_4 is

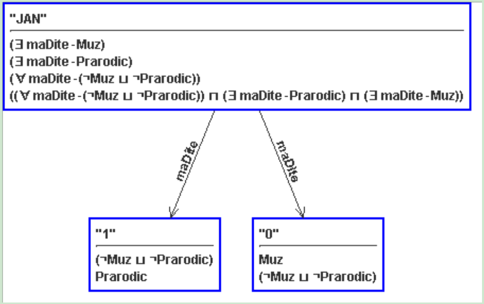


TA Run Example (2)

Example

...

- Now, four sequences of steps 2,3,4 of the TA are performed. TA state in step 4, evolves as follows:
- $\{G_0\} \xrightarrow{\sqcap\text{-rule}} \{G_1\} \xrightarrow{\exists\text{-rule}} \{G_2\} \xrightarrow{\exists\text{-rule}} \{G_3\} \xrightarrow{\forall\text{-rule}} \{G_4\}$, where G_4 is



"JAN"

(∃ maDite-Muz)
(∃ maDite-Prarodic)
(∀ maDite-(¬Muz ⊔ ¬Prarodic))
((∀ maDite-(¬Muz ⊔ ¬Prarodic)) ⊓ (∃ maDite-Prarodic) ⊓ (∃ maDite-Muz))

"1"

(¬Muz ⊔ ¬Prarodic)
Prarodic

"0"

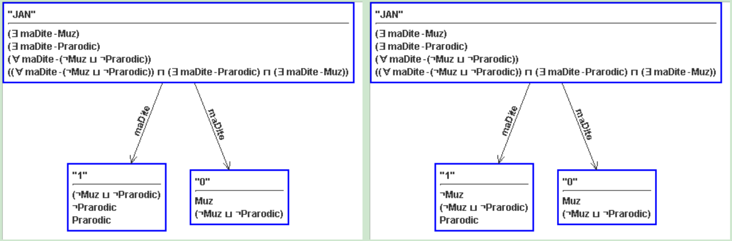
Muz
(¬Muz ⊔ ¬Prarodic)

TA Run Example (3)

Example

. . .

- By now, we applied just deterministic rules (we still have just a single completion graph). At this point no other deterministic rule is applicable.
- Now, we have to apply the $\sqcup$-rule to the concept $\neg Muz \sqcup \neg Rodic$ either in the label of node "0", or in the label of node "1". Its application e.g. to node "1" we obtain the state $\{G_5, G_6\}$ (G_5 left, G_6 right)

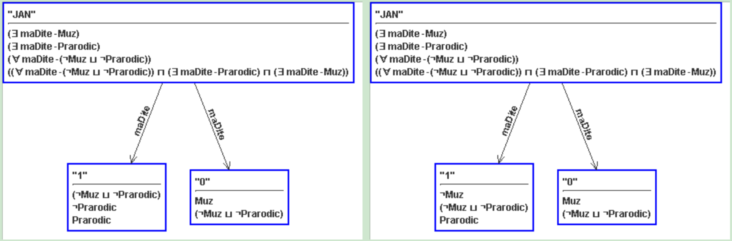


TA Run Example (3)

Example

…

- By now, we applied just deterministic rules (we still have just a single completion graph). At this point no other deterministic rule is applicable.
- Now, we have to apply the $\sqcup$-rule to the concept $\neg Muz \sqcup \neg Rodic$ either in the label of node “0”, or in the label of node “1”. Its application e.g. to node “1” we obtain the state $\{G_5, G_6\}$ (G_5 left, G_6 right)

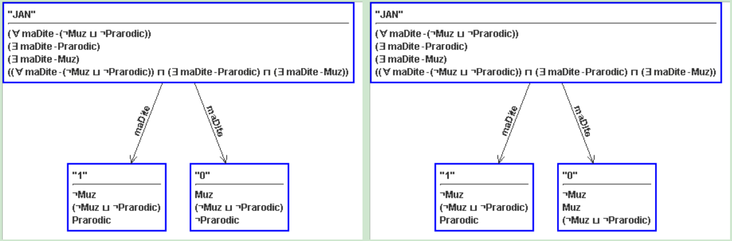


TA Run Example (4)

Example

. . .

- We see that G_5 contains a direct clash in node “1”. The only other option is to go through the graph G_6. By application of $\sqcup$-rule we obtain the state $\{G_5, G_7, G_8\}$, where G_7 (left), G_8 (right) are derived from G_6 :



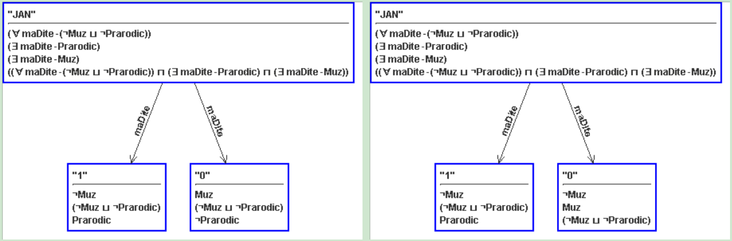
- G_7 is complete and without direct clash.

TA Run Example (4)

Example

. . .

- We see that G_5 contains a direct clash in node “1”. The only other option is to go through the graph G_6. By application of $\sqcup$-rule we obtain the state $\{G_5, G_7, G_8\}$, where G_7 (left), G_8 (right) are derived from G_6 :



- G_7 is complete and without direct clash.

TA Run Example (5)

Example

. . . A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $“JAN”^{\mathcal{I}_2} = Jan$, $“0”^{\mathcal{I}_2} = i_2$, $“1”^{\mathcal{I}_2} = i_1$,

TA Run Example (5)

Example

. . . A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $\text{“}JAN\text{”}^{\mathcal{I}_2} = Jan$, $\text{“}0\text{”}^{\mathcal{I}_2} = i_2$, $\text{“}1\text{”}^{\mathcal{I}_2} = i_1$,

TA Run Example (5)

Example

. . . A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $``JAN''^{\mathcal{I}_2} = Jan$, $``0''^{\mathcal{I}_2} = i_2$, $``1''^{\mathcal{I}_2} = i_1$,

TA Run Example (5)

Example

… A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $\text{“}JAN\text{''}^{\mathcal{I}_2} = Jan$, $\text{“}0\text{''}^{\mathcal{I}_2} = i_2$, $\text{“}1\text{''}^{\mathcal{I}_2} = i_1$,

TA Run Example (5)

Example

. . . A canonical model $\mathcal{I}_2$ can be created from G_7. Is it the only model of $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,
- $Muz^{\mathcal{I}_2} = \{i_2\}$,
- $``JAN''^{\mathcal{I}_2} = Jan$, $``0''^{\mathcal{I}_2} = i_2$, $``1''^{\mathcal{I}_2} = i_1$,

General Inclusions

We have presented the tableau algorithm for consistency checking of $\mathcal{K} = (\emptyset, \mathcal{A})$. How the situation changes when $\mathcal{T} \neq \emptyset$?

- consider $\mathcal{T}$ containing axioms of the form $C_i \sqsubseteq D_i$ for $1 \leq i \leq n$. Such $\mathcal{T}$ can be transformed into a single axiom

$$\top \sqsubseteq \top_C$$

  where $\top_C$ denotes a concept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$
- for each model $\mathcal{I}$ of the theory $\mathcal{K}$, each element of $\Delta^{\mathcal{I}}$ must belong to the interpretation of the concept at the right-hand side. How to achieve this ?

General Inclusions

We have presented the tableau algorithm for consistency checking of $\mathcal{K} = (\emptyset, \mathcal{A})$. How the situation changes when $\mathcal{T} \neq \emptyset$?

- consider $\mathcal{T}$ containing axioms of the form $C_i \sqsubseteq D_i$ for $1 \leq i \leq n$. Such $\mathcal{T}$ can be transformed into a single axiom

  $$\top \sqsubseteq \top_C$$

  where $\top_C$ denotes a concept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$
- for each model $\mathcal{I}$ of the theory $\mathcal{K}$, each element of $\Delta^{\mathcal{I}}$ must belong to the interpretation of the concept at the right-hand side. How to achieve this ?

General Inclusions

We have presented the tableau algorithm for consistency checking of $\mathcal{K} = (\emptyset, \mathcal{A})$. How the situation changes when $\mathcal{T} \neq \emptyset$?

- consider $\mathcal{T}$ containing axioms of the form $C_i \sqsubseteq D_i$ for $1 \leq i \leq n$. Such $\mathcal{T}$ can be transformed into a single axiom

$$\top \sqsubseteq \top_C$$

  where $\top_C$ denotes a concept $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$
- for each model $\mathcal{I}$ of the theory $\mathcal{K}$, each element of $\Delta^{\mathcal{I}}$ must belong to the interpretation of the concept at the right-hand side. How to achieve this ?

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

if $\top_C \notin L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

Example

Consider $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Then $\top_C$ is $\neg Muz \sqcup \exists maRodice \cdot Muz$. Let's use the introduced TA enriched by $\rightarrow_{\sqsubseteq}$ rule. Repeating several times the application of rules $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ to G_7 (that is not complete w.r.t. to $\rightarrow_{\sqsubseteq}$ rule) from the previous example we get ...

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

if $\top_C \notin L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

Example

Consider $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Then $\top_C$ is $\neg Muz \sqcup \exists maRodice \cdot Muz$. Let's use the introduced TA enriched by $\rightarrow_{\sqsubseteq}$ rule. Repeating several times the application of rules $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ to G_7 (that is not complete w.r.t. to $\rightarrow_{\sqsubseteq}$ rule) from the previous example we get …

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

if $\top_C \notin L_G(a)$ for some $a \in V_G$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

Example

Consider $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Then $\top_C$ is $\neg Muz \sqcup \exists maRodice \cdot Muz$. Let's use the introduced TA enriched by $\rightarrow_{\sqsubseteq}$ rule. Repeating several times the application of rules $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ to G_7 (that is not complete w.r.t. to $\rightarrow_{\sqsubseteq}$ rule) from the previous example we get . . .

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

if $\top_C \notin L_G(a)$ for some $a \in V_G$.

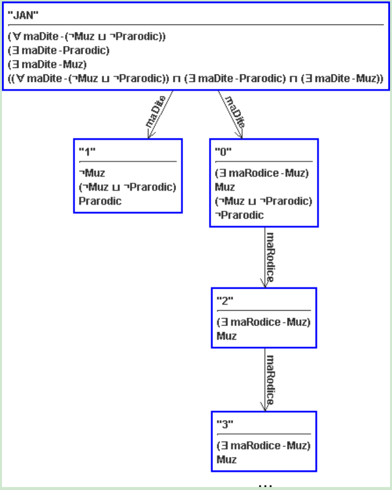
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

Example

Consider $\mathcal{K}_3 = (\{Muz \sqsubseteq \exists maRodice \cdot Muz\}, \mathcal{A}_2)$. Then $\top_C$ is $\neg Muz \sqcup \exists maRodice \cdot Muz$. Let's use the introduced TA enriched by $\rightarrow_{\sqsubseteq}$ rule. Repeating several times the application of rules $\rightarrow_{\sqsubseteq}$, $\rightarrow_{\sqcup}$, $\rightarrow_{\exists}$ to G_7 (that is not complete w.r.t. to $\rightarrow_{\sqsubseteq}$ rule) from the previous example we get . . .

General Inclusions (3)

Example



Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
  - In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.
- All inference rules are applicable until the node a in their definition is not blocked by another node.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
    - In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.
- All inference rules are applicable until the node a in their definition is not blocked by another node.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat “sufficiently frequently”.
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
  - **In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.**
- All inference rules are applicable until the node *a* in their definition is not blocked by another node.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
    - **In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.**
- All inference rules are applicable until the node a in their definition is not blocked by another node.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
  - **In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.**
- All inference rules are applicable until the node a in their definition is not blocked by another node.

Blocking in TA

- TA tries to find an infinite model. It is necessary to force it representing an infinite model by a finite completion graph.
- The mechanism that enforces finite representation is called *blocking*.
- Blocking ensures that inference rules will be applicable until their changes will not repeat "sufficiently frequently".
- For $\mathcal{ALC}$ it can be shown that so called *subset blocking* is enough:
  - **In completion graph G a node x (not present in ABOX $\mathcal{A}$) is blocked by node y, if there is an oriented path from y to x and $L_G(x) \subseteq L_G(y)$.**
- All inference rules are applicable until the node a in their definition is not blocked by another node.

Blocking in TA (2)

- In the previous example, the blocking ensures that node “2” is blocked by node “0” and no other expansion occurs. *Which model corresponds to such graph ?*
- Introduced TA with subset blocking is sound, complete and finite decision procedure for $\mathcal{ALC}$.

Blocking in TA (2)

- In the previous example, the blocking ensures that node “2” is blocked by node “0” and no other expansion occurs. *Which model corresponds to such graph ?*
- **Introduced TA with subset blocking is sound, complete and finite decision procedure for $\mathcal{ALC}$.**

Let’s play . . .

- http://krizik.felk.cvut.cz/km/dl/index.html