Description Logics – Reasoning

Petr Křemen

petr.kremen@fel.cvut.cz

November 30, 2017

Outline

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 $C_{(i)}$, we want to decide whether

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

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 $C_{(i)}$, we want to decide whether

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

(subsumption) concept C_1 *subsumes* concept C_2, i.e. $\mathcal{T} \models C_2 \sqsubseteq C_1$?

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 $C_{(i)}$, we want to decide whether

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

(subsumption) concept C_1 *subsumes* concept C_2, i.e. $\mathcal{T} \models C_2 \sqsubseteq C_1$?

(equivalence) two concepts C_1 and C_2 are *equivalent*, i.e. $\mathcal{T} \models C_1 \equiv C_2$?

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 $C_{(i)}$, we want to decide whether

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

(subsumption) concept C_1 *subsumes* concept C_2, i.e. $\mathcal{T} \models C_2 \sqsubseteq C_1$?

(equivalence) two concepts C_1 and C_2 are *equivalent*, i.e. $\mathcal{T} \models C_1 \equiv C_2$?

(disjoint) two concepts C_1 and C_2 are *disjoint*, i.e. $\mathcal{T} \models C_1 \sqcap C_2 \sqsubseteq \bot$?

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 $C_{(i)}$, we want to decide whether

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

(subsumption) concept C_1 *subsumes* concept C_2, i.e. $\mathcal{T} \models C_2 \sqsubseteq C_1$?

(equivalence) two concepts C_1 and C_2 are *equivalent*, i.e. $\mathcal{T} \models C_1 \equiv C_2$?

(disjoint) two concepts C_1 and C_2 are *disjoint*, i.e. $\mathcal{T} \models C_1 \sqcap C_2 \sqsubseteq \bot$?

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

Reducting Subsumption to Unsatisfiability

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.

$$
\begin{aligned}
(\mathcal{T} \models C_1 \sqsubseteq C_2) & & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \quad \mathcal{I} \models C_1 \sqsubseteq C_2) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \quad C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}) & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \quad C_1^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus C_2^{\mathcal{I}}) \subseteq \emptyset & \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies & \quad \mathcal{I} \models C_1 \sqcap \neg C_2 \sqsubseteq \bot & \text{iff} \\
(\mathcal{T} \models C_1 \sqcap \neg C_2 \sqsubseteq \bot)
\end{aligned}
$$

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ we want to decide whether

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ we want to decide whether

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

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ 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) $\mathcal{T} \cup \mathcal{A} \models C(a)$?

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ 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) $\mathcal{T} \cup \mathcal{A} \models R(a_1, a_2)$?

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ 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) $\mathcal{T} \cup \mathcal{A} \models R(a_1, a_2)$?

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

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ 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) $\mathcal{T} \cup \mathcal{A} \models R(a_1, a_2)$?

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

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

Inference Problems for ABOX

... and for ABOX $\mathcal{A}$, axiom α, concept C, role R and individuals $a_{(i)}$ 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) $\mathcal{T} \cup \mathcal{A} \models R(a_1, a_2)$?

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

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 ?

Reduction of concept unsatisfiability to theory consistency

Example

Consider an $\mathcal{ALC}$ theory $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a concept C and a fresh individual a_f not occuring in $\mathcal{K}$:

$$
\begin{aligned}
(\mathcal{T} \models C \sqsubseteq \bot) & \quad \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies \mathcal{I} \models C \sqsubseteq \bot) & \quad \text{iff} \\
(\forall \mathcal{I})(\mathcal{I} \models \mathcal{T} \implies C^{\mathcal{I}} \subseteq \emptyset) & \quad \text{iff} \\
\neg \left[(\exists \mathcal{I})(\mathcal{I} \models \mathcal{T} \wedge C^{\mathcal{I}} \not\subseteq \emptyset) \right] & \quad \text{iff} \\
\neg \left[(\exists \mathcal{I})(\mathcal{I} \models \mathcal{T} \wedge a_f^{\mathcal{I}} \in C^{\mathcal{I}}) \right] & \quad \text{iff} \\
(\mathcal{T}, \{C(a_f)\}) \quad \text{is inconsistent} &
\end{aligned}
$$

Note that for more expressive description logics than $\mathcal{ALC}$, the ABOX has to be taken into account as well due to its interaction with TBOX.

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 [**dlh2003**].

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [**dlh2003**].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [**dlh2003**].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite

other ... – e.g. resolution-based, transformation to finite automata, etc.

Inference Algorithms in Description Logics

Structural Comparison is polynomial, but complete just for some simple DLs *without full negation*, e.g. $\mathcal{ALN}$, see [**dlh2003**].

Tableaux Algorithms represent the State of Art for complex DLs – sound, complete, finite

other ... – e.g. resolution-based, transformation to finite automata, etc.

We will introduce tableau algorithms.

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.” (resp. theory $\mathcal{K}$)**

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$." (resp. theory $\mathcal{K}$)**
- Each TA can be seen as a *production system* :

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$." (resp. theory $\mathcal{K}$)**
- 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),

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$." (resp. theory $\mathcal{K}$)**
- 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

Tableaux Algorithms

- Tableaux Algorithms (TAs) serve for checking theory consistency in a simple manner: **"Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$." (resp. theory $\mathcal{K}$)**
- 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 theory consistency in a simple manner: **“Consistency of the given ABOX $\mathcal{A}$ w.r.t. TBOX $\mathcal{T}$ (resp. consistency of theory $\mathcal{K}$) is proven if we succeed in constructing a model of $\mathcal{T} \cup \mathcal{A}$.” (resp. theory $\mathcal{K}$)**
- 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
- TAs are not new in DL – they were known for FOL as well.

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 $L_G(\langle x, y \rangle)$[1]

[1] 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 $L_G(\langle x, y \rangle)$[1]

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 $x \in V_G$

[1]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 $L_G(\langle x, y \rangle)$[1]

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.

[1] 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 $L_G(\langle x, y \rangle)$[1]

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.

[1] 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

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

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
- $R(a_1, a_2)$ for each edge $\langle a_1, a_2 \rangle \in E_G$ and each role $R \in L_G(a_1, a_2)$

Tableau Algorithm for $\mathcal{ALC}$

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

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

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. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

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. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

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:

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. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

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 $C(a) \in \mathcal{A}$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$

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. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

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 $C(a) \in \mathcal{A}$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
- for each $R(a_1, a_2) \in \mathcal{A}$ put $\langle a_1, a_2 \rangle \in E_{G_0}$ and $R \in L_{G_0}(a_1, a_2)$

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. $\neg(C_1 \sqcap C_2)$ is equivalent (de Morgan rules) to $\neg C_1 \sqcup \neg C_2$).

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 $C(a) \in \mathcal{A}$ put $a \in V_{G_0}$ and $C \in L_{G_0}(a)$
- for each $R(a_1, a_2) \in \mathcal{A}$ put $\langle a_1, a_2 \rangle \in E_{G_0}$ and $R \in L_{G_0}(a_1, a_2)$
- 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"

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”

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

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 $a \in V_G$.

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.

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

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 $a \in V_G$.

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.

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

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 $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

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

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ 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 $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{a_2\}, E_G \cup \{\langle a_1, a_2\rangle\}, L_{G'})$, a $L_{G'}(a_2) = \{C\}$, $L_{G'}(a_1, a_2) = \{R\}$ 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 $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{a_2\}, E_G \cup \{\langle a_1, a_2\rangle\}, L_{G'})$, a $L_{G'}(a_2) = \{C\}$, $L_{G'}(a_1, a_2) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

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 $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{a_2\}, E_G \cup \{\langle a_1, a_2 \rangle\}, L_{G'})$, a $L_{G'}(a_2) = \{C\}$, $L_{G'}(a_1, a_2) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_\forall$ rule

if $(\forall R \cdot C) \in L_G(a_1)$ and there exists $a_2 \in V_G$ such that $R \in L_G(a, a_1)$ and at the same time $C \notin L_G(a_2)$.

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

$\rightarrow_{\sqcap}$ rule

if $(C_1 \sqcap C_2) \in L_G(a)$ and $\{C_1, C_2\} \nsubseteq L_G(a)$ 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 $(\exists R \cdot C) \in L_G(a_1)$ and there exists no $a_2 \in V_G$ such that $R \in L_G(a_1, a_2)$ and at the same time $C \in L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G \cup \{a_2\}, E_G \cup \{\langle a_1, a_2 \rangle\}, L_{G'})$, a $L_{G'}(a_2) = \{C\}$, $L_{G'}(a_1, a_2) = \{R\}$ and otherwise is the same as L_G.

$\rightarrow_{\forall}$ rule

if $(\forall R \cdot C) \in L_G(a_1)$ and there exists $a_2 \in V_G$ such that $R \in L_G(a, a_1)$ and at the same time $C \notin L_G(a_2)$.

then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, and $L_{G'}(a_2) = L_G(a_2) \cup \{C\}$ and otherwise is the same as L_G.

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

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

TA Run Example (2)

Example

. . .

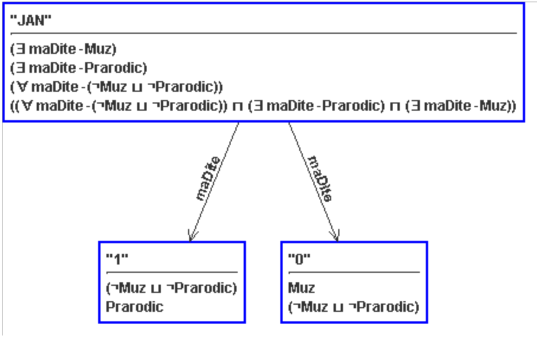
- Now, four sequences of steps 2,3,4 of the TA are performed. TA state in step 4, evolves as follows:

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 (3)

Example

...

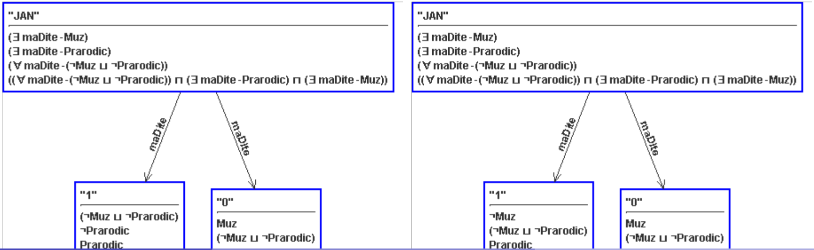
- By now, we applied just deterministic rules (we still have just a single completion graph). At this point no other deterministic rule is applicable.

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)

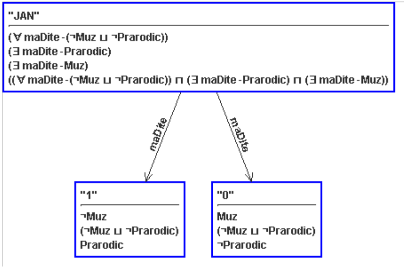


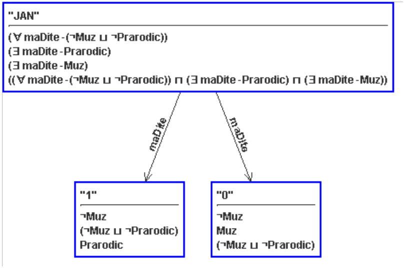
TA Run Example (4)

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 ⊔-rule we obtain the state $\{G_5, G_7, G_8\}$, where G_7 (left), G_8 (right) are derived from G_6 :



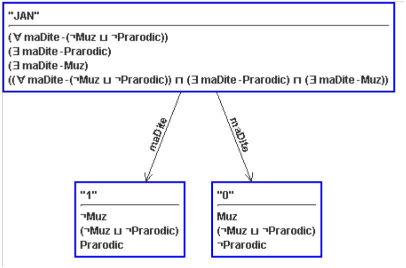


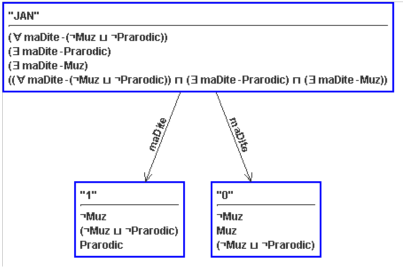
TA Run Example (4)

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 ⊔-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 $\mathcal{K}_2$?

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,

TA Run Example (5)

Example

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

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1 \rangle, \langle Jan, i_2 \rangle\}$,

TA Run Example (5)

Example

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

- $\Delta^{\mathcal{I}_2} = \{Jan, i_1, i_2\}$,
- $maDite^{\mathcal{I}_2} = \{\langle Jan, i_1\rangle, \langle Jan, i_2\rangle\}$,
- $Prarodic^{\mathcal{I}_2} = \{i_1\}$,

TA Run Example (5)

Example

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

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

TA Run Example (5)

Example

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

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

Finiteness

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

- $\mathcal{K}$ is 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.

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 $\mathcal{A}$.

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:

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
  - As a result $a_1^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
  - As a result $a_1^\mathcal{I} \in (\exists R \cdot C)^\mathcal{I}$.
  - Next, $i \in \Delta^\mathcal{I}$ must exist such that $\langle a_1^\mathcal{I}, i \rangle \in R^\mathcal{I}$ and at the same time $i \in C^\mathcal{I}$.

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
    - As a result $a_1^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
    - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a_1^{\mathcal{I}}, i \rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
    - By application of $\rightarrow_\exists$ a new node a_2 was created in G_{i+1} and the label of edge $\langle a_1, a_2 \rangle$ and node a_2 has been adjusted.

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
  - As a result $a_1^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
  - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a_1^{\mathcal{I}}, i \rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
  - By application of $\rightarrow_\exists$ a new node a_2 was created in G_{i+1} and the label of edge $\langle a_1, a_2 \rangle$ and node a_2 has been adjusted.
  - It is enough to place $i = a_2^{\mathcal{I}}$ 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.

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, $(\exists R \cdot C) \in L_{G_i}(a_1)$ held for $a_1 \in V_{G_i}$.
  - As a result $a_1^{\mathcal{I}} \in (\exists R \cdot C)^{\mathcal{I}}$.
  - Next, $i \in \Delta^{\mathcal{I}}$ must exist such that $\langle a_1^{\mathcal{I}}, i \rangle \in R^{\mathcal{I}}$ and at the same time $i \in C^{\mathcal{I}}$.
  - By application of $\rightarrow_\exists$ a new node a_2 was created in G_{i+1} and the label of edge $\langle a_1, a_2 \rangle$ and node a_2 has been adjusted.
  - It is enough to place $i = a_2^{\mathcal{I}}$ 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.

- 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 $A^{\mathcal{I}} = \{a \mid A \in L_G(a)\}$
- 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 $R^{\mathcal{I}} = \{\langle a_1, a_2 \rangle \mid R \in L_G(a_1, a_2)\}$
- 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 ?

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.

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 ?

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 ?
  - P-SPACE (between NP and EXP-TIME).

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

$$\top \sqsubseteq \top_C$$

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 $(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$

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 $\top_C^{\mathcal{I}}$. How to achieve this ?

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

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

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.

General Inclusions (2)

What about this ?

$\rightarrow_{\sqsubseteq}$ rule

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

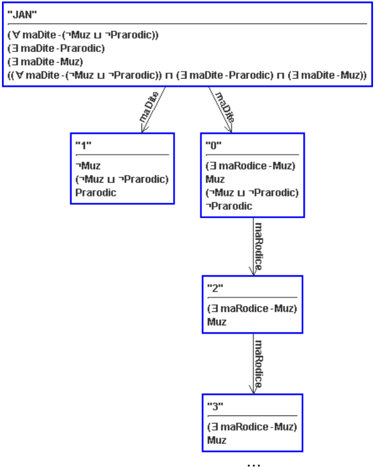
then $S' = S \cup \{G'\} \setminus \{G\}$, where $G' = (V_G, E_G, L_{G'})$, a $L_{G'}(a) = L_G(a) \cup \{\top_C\}$ and otherwise is the same as L_G.

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



. . . this algorithm doesn't necessarily terminate ☹.

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.

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 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".

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:

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 $L_G(x) \subseteq L_G(y)$.**

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 $L_G(x) \subseteq L_G(y)$.**
- *exists*$-$ rule is only applicable if the node a_1 in its 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 ?*

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://kbss.felk.cvut.cz/tools/dl

References I

[1] * Vladimír Mařík, Olga Štěpánková, and Jiří Lažanský.
Umělá inteligence 6 [in czech], Chapters 2-4.
Academia, 2013.

[2] * Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors.
The Description Logic Handbook, Theory, Implementation and Applications, Chapters 2-4.
Cambridge, 2003.

[3] * Enrico Franconi.
Course on Description Logics.
http://www.inf.unibz.it/ franconi/dl/course/, cit. 22.9.2013.