University of Ostrava

Institute for Research and Applications of Fuzzy Modeling

Reasoning in Description Logic with Semantic Tableau Binary Trees

Alena Lukasová

Research report No. 63

2005

Submitted/to appear:

Description Logic Workshop 2005, Edinburg

Supported by:

Research project MSM 6198898701 of Czech Ministry of education

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
Bráfova 7, 701 03 Ostrava 1, Czech Republic

tel.: +420-69-6160234 fax: +420-69-6120 478
e-mail: Alena.Lukasova@osu.cz

Reasoning in Description Logic with Semantic Tableau Binary Trees [*]

Alena Lukasová
University of Ostrava, Ostrava, Czech Republic
alena.lukasova@osu.cz

Abstract

An alternative version of the tableau decision making algorithm defined in the "Handbook of Description Logics" (Baader et col.) is presented. It is a version of semi-decidable tableau formal system that corresponds with that one of first order logic and shares its semantic soundness and completeness. In examples the advantages of the transparent tree-style representation algorithm, comparing with the algorithm of the handbook, is demonstrated. It is shown that the tree representation of the tableau proof gives a possibility of transparent creation of models in the cases of direct tableau proofs. On the base of the properties of indirect tableau proofs is possible to define a dual Gentzen-like axiomatic system for direct generation of theories consisting of logical consequences of knowledge bases.

Key words: Description logic, semantic tableau

Introduction

It seams that the tableau decision making algorithm defined in the "Handbook of Description Logics" [Ba03] has been generally accepted by the description logic community. Even if it has advantages in the properties of finality (decidability) we would like to point out some of its inabilities or failings. After defining of language and its semantics we'll define our version of semi-decidable tableau formal system that corresponds with that one of first order logic ([Lu03]) and shares its semantic soundness and completeness. We enumerate cases of the algorithm usage and show them in examples. We also point out in examples the advantages of our transparent tree-style representation algorithm compared with

[*]This work was supported by grant MSM 6198898701 of Czech Ministry of education

the algorithm of the handbook [Ba03]. We'll show that the tree representation of the tableau proof gives us a possibility of transparent creation of models in the cases of direct tableau proofs. On the base of the properties of indirect tableau proofs we show a possibility to define a dual Gentzen-like axiomatic system for direct generation of theories consisting of logical consequences of knowledge bases. Generally we introduce an axiomatic system by

1. specifying a language for representation of knowledge concerning the special reference system W,
2. defining logical axioms that are logical valid formulas,
3. writing a knowledge base Σ representing relevant knowledge on W as a set of special axioms (basic knowledge) of the theory,
4. defining an inference mechanism (set of rules) for building theories.

There are some special details in the frame of our approach concerning the four points enumerated above that we have to make clear more precise.

1 Language L_{DL} - formal definitions of syntax and semantics

The language L_{DL} of description logic (DL) we introduce here by the Backus-Naur form is a special ALC(Q)-language. DL expressions represent knowledge about a reference system W (world to be represented) by means of concepts and roles they play in the context of the system. The "Q" in brackets means that it is possible, if necessary, to add number restriction constructors $\exists^{>n}R$ and $\exists^{\geq n}R$ to the L_{DL}.

1.1 Specifying the language L_{DL} of DL

Defining a language L_{DL}, we distinguish between formulas of the language and meta-language schemes. For individual objects names we don't assume the "unique name assumption".

Definition 1 *(L_{DL} grammar rules) DLformulas of a language L_{DL} are concept formulas (Cformulas) or role formulas (Rformulas) or their instances, both* *created by a special set of DL constructors defined as follows:*

$$
\begin{aligned}
< term > &::=< individual > \\
&::=< variable > \\
< individual > &::= a/b/a1/\ldots \\
< variable > &::= x/y/x1/\ldots \\
< atomic\ role > &::= R/S/\ldots \\
< Rformula > &::=< atomic\ role > \\
&::= (< Rformula >) \sqcap (< Rformula >) \\
&::= \neg(< Rformula >) \\
< role\ instance > &::=< Rformula > (< term >, < term >) \\
< atomic\ concept > &::= A/B/\ldots \\
< Cformula > &::=< atomic\ concept > \\
&::= \top \\
&::= \bot \\
&::= \neg(< Cformula >) \\
&::= (< Cformula >) \sqcap (< Cformula >) \\
&::= (< Cformula >) \sqcup (< Cformula >) \\
&::= \exists(< Rformula >).\top \\
&::= \exists(< Rformula >).(< Cformula >) \\
&::= \forall(< Rformula >).(< Cformula >) \\
< concept\ instance > &::=< Cformula > (< term >)
\end{aligned}
$$

(As the unary predicate symbols will represent concepts and binary predicate symbols will represent roles, it isn't necessary to write in the definitions of concept and role expressions (atomic or composed) with explicit determinations of n-arities of predicate symbols.)

1.2 Interpretation of DLformulas

Definition 2 *(interpretation structure and rule)Interpretation structure of the language L_{DL} is a pair $I = (\Delta^I, \bullet^I)$, where the set Δ^I is a universe of discourse, $\bullet^I$ is a denotation function that maps a Cformula C into it's extent $C^I \subseteq \Delta^I$ and Rformula R into it's extent $R^I \subseteq \Delta^I \times \Delta^I$. For concepts C, D, roles R, S, individuals c, d_1, d_2 and term t hold the following interpretation rules :*

Rformula interpretation rules:

$$R^I = \{(d1, d2) \mid R(d1, d2)\}$$
$$(R \sqcap S)^I = R^I \cap S^I$$
$$(\neg R)^I = (\Delta^I \times \Delta^I) \backslash R^I$$
$$(R(a, b))^I = \{(a, b)\} \in R^I$$

Cformula interpretation rules:

$$C^I = \{c \mid C(c)\}$$
$$\top^I = \Delta^I$$
$$\bot^I = \emptyset$$

$$(\neg C)^I = \Delta^I \backslash C^I$$
$$(C \sqcap D)^I = C^I \cap D^I$$
$$(C \sqcup D)^I = C^I \cup D^I$$
$$(\forall R.C)^I = \{d_1 \in \Delta^I \mid \forall d_2 : (d_1, d_2) \in R^I \to d_2 \in C^I\}$$
$$(\exists R.\top)^I = \{d_1 \in \Delta^I \mid \exists d_2 : (d_1, d_2) \in R^I\}$$
$$(\exists R.C)^I = \{d_1 \in \Delta^I \mid \exists d_2 : (d_1, d_2) \in R^I \wedge d_2 \in C^I\}$$
$$(C(a))^I = \{a\} \in C^I$$

1.3 Concept consistency, subsumption and taxonomy

Definition 3 *(consistency of Rformula/Cformula in an interpretation) Rformula/Cformula is consistent in the interpretation I of the language L_{DL} iff its extent is nonempty in the interpretation I.*

Definition 4 *(concept subsumption) Let C, D be Cformulas of a DL language L_{DL} . C is subsumed by D, expressed by $C \sqsubseteq D$, iff $C^I \subseteq D^I$ for every interpretation I of the language L_{DL}, C is equivalent D, expressed by $C \equiv D$, iff $C^I \equiv D^I$ for every interpretation I of the language L_{DL}.*

2 Knowledge base in DL

Definition 5 *(knowledge base) Given DL language L_{DL} for representation of a referent system (world) W, knowledge base (KB) S is a pair $S = (T, A)$, where T is the TBox of terminological axioms representing intensional knowledge about W and A is the ABox of assertions representing some extensional knowledge about W.*

2.1 TBox and ABox of knowledge base

Definition 6 *(axioms of TBox and ABox) Let C be a concept name, R be a role name, t, s are names of terms of the L_{DL} . Terminological axioms of TBox are meta-language expressions that define concepts by Cformulas (tab.1, expressions (1) - (3)). ABox (assertional box) is a box of DLformulas of a language L_{DL} representing extensional knowledge of knowledge base. The DLformulas of ABox are concept instances or role instances (def. 1) having the forms of expressions (4) - (5) (tab.1).*

Expression (1) of the tab.1 says that an atomic concept belongs to the special concept taxonomy (hierarchy, ontology) we treat within the referent system. Definitions or specifications of concepts of TBoxes are meta-formulas. So, if they ought to appear in formal proof sequences, they have to be remade into DLformulas. We can construct DLformulas of ABox of the KB Σ using individual object names or terms only if we previously have defined the corresponding TBox concept or role symbols to be used within the ABox of the KB Σ .

Table 1: Syntax and sematics of TBox and ABox expressions

	Type of TBox expression	Syntax	Semantics
(1)	atomic concept specification	$A \sqsubseteq T$	$A^I \subseteq \Delta^I$
(2)	concept definition	$C \equiv D$	$C^I = D^I$
(3)	concept specification (subsumption)	$C \sqsubseteq D$	$C^I \subseteq D^I$
	Type of ABox expression		
(4)	Concept assertion	$C(t)$	$t^I \in C^I$
(5)	Role assertion	$R(s,t)$	$(s^I, t^I) \in R^I$

2.2 Consistency and models of knowledge bases

To be able to make decisions about consistency of a KB we have to consider the KB as a set of DLformulas. It means: to decide consistency of KB it is necessary first to rewrite TBox expressions and ABox expressions into a set of DLformulas. We needn't do anything with DLformulas of ABox but the expressions of TBox aren't DLformulas and we must remake them into the corresponding set of DLformulas with the help of the following theorem.

Theorem 1 *(consistency of TBox expressions) Let $T = C \sqsubseteq D$, ($T = C \equiv D$) be a TBox expression of a KB of a language L_{DL}, let C, D be concepts. Then T is consistent in an interpretation I iff $\neg C \sqcup D$, $((\neg C \sqcup D)(\neg D \sqcup C))$ is consistent in I.*

Proof: According to the definition of concept subsumption D subsumes C, iff $C^I \subseteq D^I$ for every interpretation I, C is equivalent D, iff $C^I \equiv D^I$, ($C^I \subseteq D^I$ and $D^I \subseteq C^I$) for every interpretation I. From the set-theory point of view we can treat the meaning of the concept subsumption $C^I \subseteq D^I$ as the union of sets $(\Delta^I \backslash C^I) \cup D^I$ corresponding in the frame of DL semantics (def. 4) to the disjunction of concepts $\neg C \sqcup D$. Similarly for the case of equivalency of concepts we have the set $((\Delta^I \backslash C^I) \cup D^I)((\Delta^I\ D^I) \cup C^I))$, corresponding to the concept conjunction $(\neg C \sqcup D) \sqcap (\neg D \sqcup C))$ in DL.

Immediately from the definition of the semantics of DLformulas (def. 4) of ABox follows that the following theorem holds.

Theorem 2 *(consistency of ABox formulas) Let $A = C(a)$, ($A = R(a,b)$) be a Cformula (Rformula) of the language L_{DL} of an ABox of KB S, I be an interpretation of the L_{DL}. A is satisfied in the interpretation I iff $a^I \in C^I$, ($(a^I, b^I) \in R^I$) holds.*

In the following text we shall use the knowledge base $\Sigma = \{TBox, ABox\}$ remade into a set S of DLformulas.

Definition 7 *(consistency and models of knowledge bases) Structure of an interpretation I is a model of TBox T, (ABox A) of a knowledge base S iff all the DLformulas created by rewriting of the TBox T (ABox A) definitions hold in I. Structure of an interpretation I is a model of a knowledge base S, iff all the DLformulas created by rewriting of the TBox T and all the formulas of ABox A of the knowledge base S hold in I.*

An important role in the frame of reasoning on knowledge bases play the DLformulas that are logical valid. Those in all interpretations logical valid formulas can become parts of any knowledge base because they don't change its set of models.

Definition 8 *(logical validity of DLformula) DLformula is logical valid iff all the applicable interpretations are its models.*

Definition 9 *(logical consequence of a KB) DLformula F is a logical consequence of a knowledge base S ($S| = F$) iff it holds in all models of the S.*

3 Reasoning on knowledge bases

DLformulas in their transformed form represent a set of special axioms for building theory in a special formal system.

We require formal tools of reasoning on KB to be able to solve the following tasks:

1. First, it is necessary for every KB to answer the question: Do the KB S have a model or S isn't a consistent set of DLformulas? If no, it has no sense in this case to reason anything from S because it would be possible to deduce from S some DLformula and its negation.

2. We require KB S to stay consistent after an adding a new knowledge T into its TBox. In this case, solving the problem consists in decision of the consistency of the set $S \cup T$ where T represents the new expression of TBox transformed into the corresponding DLformula.

3. KB S has to stay consistent also after an adding a new knowledge A into its ABox. In this case we can add a DLformula A into the ABox iff the corresponding concepts or roles occurring in the A have been defined or specified in the TBox of the S. Open world assumption assure the possibility to add new instances of Cformulas (Rformulas).

4. We would like to decide if a DLformula F is a logical consequence of the KB Σ . If yes, we expect $S \cup \neg F$ ought to be inconsistent.

5. It would be useful for some proofs to use some convenient logical valid DLformulas. Those DLformulas proved by means of unsatisfiability of their negations can be added to any KB because their logical validity is independent on concrete interpretations.

6. We would like to have to our disposal some generator of logical consequences of the S.

All of the tasks 1 - 6 we can transform into problems of consistency (satisfiability). In the remaining part of this paper we present a formal system based on semantic tableau binary trees that may become a convenient tool for solving those tasks. We also proof its semantic soundness and completeness and show some examples of its practical using.

4 Tableau proofs

4.1 Definition of semantic tableau

Authors of the [Ba03] present tableaux algorithms by means of a collection of so-called completion rules intended to generate completion of ABox completed with respect to a corresponding TBox of a knowledge base S. An approach we present here is a bit different. We define here our tableau algorithm in the way similar to the tableau algorithm in predicate logic ([Lu03]) . We use a set S of DLformulas of a KB containing DLformulas of ABox and transcriptionsof definitions and specifications of TBox into Cformulas.

Definition 10 *For a given set S of DLformulas we define the semantic tableaux as a binary tree with labeled nodes according to the following construction rules:*

1. *Level 0: The root is labeled by the list of all DLformulas of S.*

2. *Construction of $(n+1)th$ level nodes:*
α -rule: If the label list at a level n contains a α -DLformula $(C(x) \sqcap D(x))$ which we actually have chosen for the creation of a unique $(n+1)th$ level node, we copy to it the label of the node n and add to the list both $C(x)$ and $D(x)$ DLformulas.
β -rule: If the label list at a level n contains a β -DLformula $(C(x) \sqcup D(x))$ which we actually have chosen for the creation of two following level nodes, we create two nodes at the level $n+1$, copy to each one the label of the node n and add to the list of the left node the DLformula $C(x)$ and to the right node the DLformula $D(x)$.
γ -rule: If the label list at a level n contains a γ -DLformula $\forall R.C(x)$ which we actually have chosen for the creation of an unique $(n+1)th$ level

node, we copy to it the label of the node n and add to the list DLformulas $R(x,t), \neg R(x,t) \sqcup C(t)$ for a term t of the language L_{DL}.
δ -rule: If the label list at a level n contains a δ -DLformula $\exists R.C(x))$ ($\exists R.\top$)) which we actually have chosen for the creation of an unique $(n+1)$th level node, we copy to the new node the label of the node n and add to the list DLformulas $R(x,a), C(a)$ (formula $R(x,a)$) with a new constant term a.

3. *If there exists a complementary pair of DLformulas in a label list of a node n, we close the branch by the node n as a leave that makes the corresponding branch closed.*

4. *If the condition 3 doesn't happen, then: if there is no possibility for a node n to apply rules α, β, γ, δ and to create a following level node, the node n becomes a leaf that makes the branch finitely open, in the opposite case the branch become potentially infinitely open.*

Definition 11 *Semantic tableau with all branches closed is a closed tableau.*

4.2 Soundness and completeness

Theorem 3 *(semantic soundness and completeness of the formal tableau proof) A set S of DLformulas of the language L_{DL} is inconsistent iff a closed semantic tableau of S exists.*

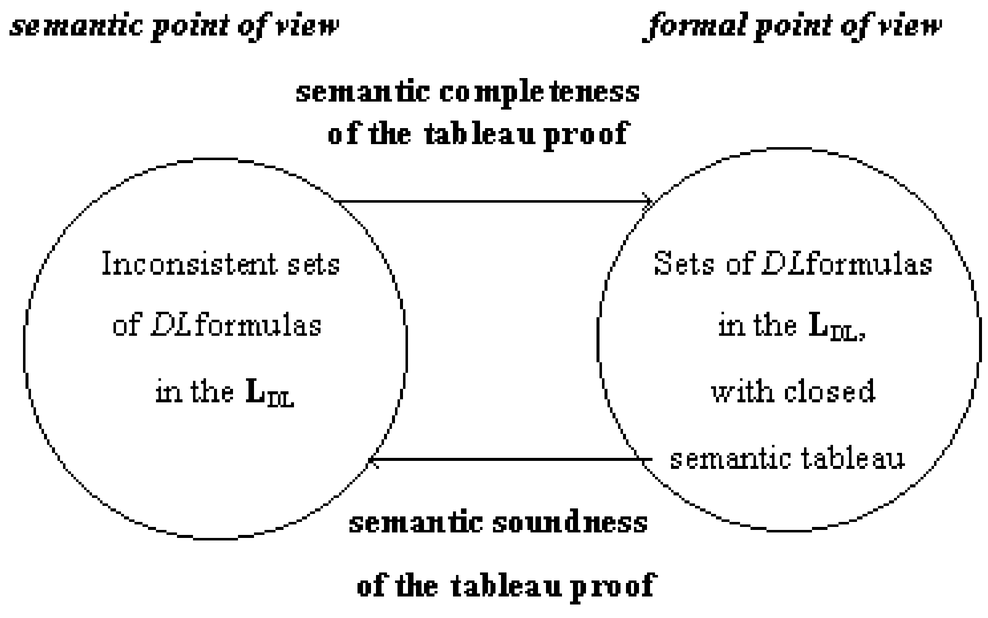

semantic point of view
formal point of view
semantic completeness of the tableau proof
Inconsistent sets of DLformulas in the $\mathbf{L}_{\mathrm{DL}}$
Sets of DLformulas in the $\mathbf{L}_{\mathrm{DL}}$, with closed semantic tableau
semantic soundness of the tableau proof

fig. 1 Semantic soundness and completeness of tableau proofs

1) Proof of the semantic soundness of the tableau algorithm is in fact a proof of the semantic soundness of the rules $\alpha, \beta, \gamma, \delta$ defined in the def.10. Proof by induction on the sub-tree depth: If a sub-tree of a depth h with a root of a level n closes then the list of DLformulas $U(n)$ in its root label is inconsistent. With $h = 0$ it is involved a leaf of the tableau. If the tableau closes the set $U(0)$ must contain a complementary pair of literals and it means that $U(0)$ is inconsistent. Let us assume the induction assumption until a depth $h - 1$: If a semantic tableau sub-tree of a depth $k < h$ closes, then the list of DLformulas in its root label is inconsistent. Now let us consider a semantic tableau sub-tree of the depth h with a root node n. To create the following node's label we need to use one of the $\alpha, \beta, \gamma, \delta$ - rules. For the use of the α -rule (β -rule, γ -rule, δ-rule) the list of DLformulas must have a form U, α ($U, \beta, U, \gamma, U, \delta$), α -DLformula (β -DLformula, γ -DLformula, δ-DLformula) has been specified in the def.10. 2a) (2b, 2c, 2d), U is a list (event. empty) of DLformulas. a) Let in the case of the α -rule the label of a node is $U(n) = U, A_1 \sqcap A_2$. Then the label of the unique successor - the node n' of the following level has according to the α -rule a form $U(n') = U, A_1, A_2$. As the depth of the node n' is $h-1$, the induction assumption holds. It means if the sub-tree of the node n' closes, the set of DLformulas of the $U(n') = U, A_1, A_2$ is inconsistent. Let us assume the sub-tree of the node n' closes. As n' is a unique successor of n, the sub-tree of the node n must close. In this case must be inconsistent the set of DLformulas $U(n) = U, A_1 \sqcap A_2$. If there would exist a set of objects that makes $U, A_1 \sqcap A_2$ consistent, the set must fulfill

also U, A_1, A_2 - that contradicts to the induction assumption. b) Let in the case of the β -rule the label of a node is $U(n) = U, B_1 \sqcup B_2$. Then the label of two successors - the nodes n' and n'' of the following level has according to the β -rule forms $U(n') = U, B_1, U(n'') = U, B_2$. As the depth of the nodes n', n'' is $h-1$, the induction assumption holds - if the sub-trees of the nodes n', n'' closes, the sets $U(n'), U(n'')$ are inconsistent. Let us assume the sub-trees of the nodes n', n'' close. As n', n'' are successors of n, the sub-trees of the node n must close and the sets $U(n')$, $U(n'')$ must be inconsistent. If there would exist a set of objects that makes $U, B_1 \sqcup B_2$ consistent, the set must fulfill also U, B_1 or U, B_2, that contradicts with the induction assumption. c) For using the γ -rule is necessary a sub-tree root label of the depth h has to be of the form $U(n) = U, \forall R.C$. Applying of the γ -rule now consists in creating a unique successor node with the label $U(n') = U, \forall R.C, R(x,t), \neg R(x,t) \sqcup C(t)$, t is a term, that belongs to the sub-tree of a depth $h-1$ fulfilling the induction assumption. So if the sub-tree closes, the set $U(n')$ of its root label is inconsistent. It is obvious that interpretation structures fulfilling $\forall R.C$ must be the same as this one fulfilling $\forall R.C, R(x,t), \neg R(x,t) \sqcup C(t)$. d) For using the δ-rule is necessary, a sub-tree root label of the depth h has to be of the form $U(n) = U, \exists R.C$. Applying of the δ-rule creates an unique successor node with the label $U(n') = U, \exists R.C, R(x,d), C(d)$, ($d$ is a new constant term) that belongs to the sub-tree of a depth $h-1$ fulfilling the induction assumption. If the sub-tree closes the set $U(n')$ of its root label is inconsistent. It is obvious that interpretation structures fulfilling $U, \exists R.C$ has to be the same as this one fulfilling $U, \exists R.C, R(x,d), C(d)$.
2) Proof of the semantic completeness: Using the indirect method we have to proof: If there is some open branch in the tableau tree, then the set S is consistent. We'll utilize properties of model sets defined as follows.

Definition 12 *(model set of DLformulas) A set U of DLformulas of the language L_{DL} is a model set if the following holds:*

1. *(1) If $C(a)$ is an instantion of a concept C then $C(a) \notin U$ or $\neg C(a) \notin U$.*
2. *(2) If $A \in U$ is a α -DLformula consisting of components A_1, A_2, then $A_1 \in U, A_2 \in U$.*
3. *(3) If $B \in U$ is a β -DLformula consisting of components B_1, B_2, then $B_1 \in U$ or $B_2 \in U$.*
4. *(4) If $A \in U$ is a γ-DLformula of the form $\forall R.C$, then $R(x,t) \in U, \neg R(x,t) \sqcup C(t) \in U$ (t is a term).*
5. *(5) If $A \in U$ is a δ-DLformula of the form $\exists R.C$, then $R(x,d) \in U, C(d) \in U$ for a object d of the universe W.*

Lemma 1 *Let V be an open branch of a semantic tableau of a DLformulas set S. Then a set $U = \Sigma_{i=1,..n}U(i)$ unifying all label lists along the branch V represents a model set.*

Proof: Applying tableau rules of the def. 10 guaranties that (2)-(5) holds. (1) holds because for an open branch holds $C(a) \in U$ or $\neg C(a) \in U$, not both because the branch is open.

Lemma 2 *If U is a model set, it is a consistent one.*

Proof: For the model set U of DLformulas we define an interpretation of $C(a)$ in the following way: If $C(a) \in U$, then $a^I \in C^I$, if $\neg C(a) \in U$, then $a^I \in \neg C^I$. Consistency of $C(a), \neg C(a)$ guaranties the fact that only one item of the pair occurs in U. We'll proof by induction on complexity of DLformula $A \in U$ that in the case of the interpretation defined above A must be consistent. 1. For $n = 1$ is A consistent because in the case of $A = C(a)$ C(a)holds , in the case of $A = \neg C(a)$ $\neg C(a)$holds. 2.

1. a) If A is a α -DLformula $A_1 \sqcap A_2$ of complexity n, then according to (2)is $A_1 \in U$ and $A_2 \in U, A_1, A_2$ are formulas of complexity $n-1$ fulfilling the induction assumption $A_1^I \neq \oslash$ a $A_2^I \neq \oslash$. If $A_1 \sqcap A_2$ were inconsistent at least one of DLformulas A_1, A_2 would have to be inconsistent contrary to the induction assumption.

2. b) If A is a β -formula $A_1 \sqcup A_2$ of complexity n, then according to (3) is $A_1 \in U$ or $A_2 \in U, A1, A2$ of the complexity $n-1$ fulfill the induction assumption $A_1^I \neq \oslash$ a $A_2^I \neq \oslash$. If $A_1 \sqcup A_2$ were inconsistent, both DLformulas A_1, A_2 would be inconsistent contrary to the induction assumption .

3. c) If A is a γ -DLformula $\forall R.C$ of the complexity n, then according to (4) besides $A \in U$ also $R(x,t) \in U, \neg R(x,t) \sqcup C(t) \in U$ holds, t is a term occurring in U. If $\forall R.C$ were inconsistent both $R(x,t)$ and $C(t)$ wouldn't hold, contrary to the induction assumption.

4. d) If A is δ -DLformula $R.C$ of the complexity n, then according to (5) besides $A \in U$ also $R(x,d) \in U, C(d) \in U$ holds, d is a new constant term not occurring until now in U. If $R.C$ were inconsistent both $R(x,d)$ and $C(d)$ wouldn't hold contrary to the induction assumption.

Theorem 4 *(completeness) If A is a logical valid DLformula of the language LDL then a semantic tableau of $\neg A$ closes.*

Proof: Completeness of the semantic tableau formal system follows immediately from the previous lemma. Open branch of the tableau induces a model set U so that a model exists. The model fulfill also the DLformula A because $A \in U$.

4.3 Using tableau proofs

The aim of the following examples is to show how to use direct or undirect tableau proofs for solving the problems named in the previous parahraph. If it would be usefull we shall use besides the tableau rules (def. 10) the following valid equivalencies:

$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$$

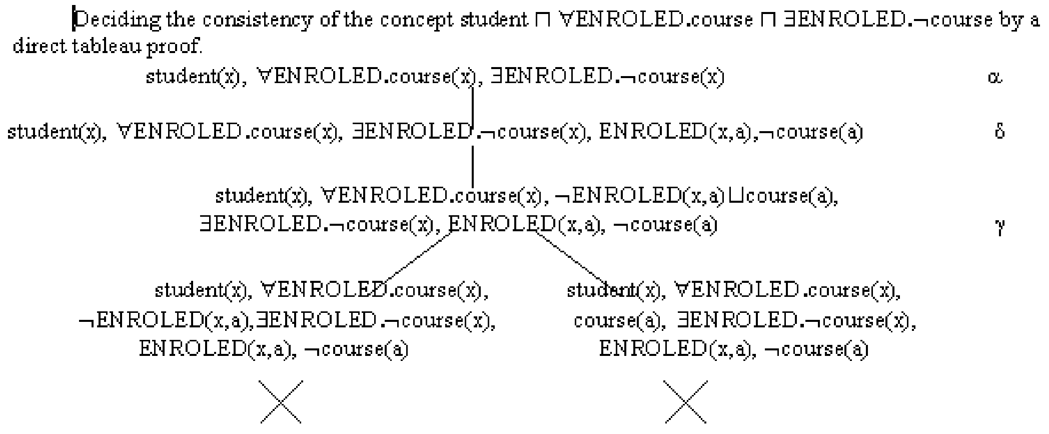
$$(C \sqcap D) \equiv \neg C \sqcup \neg D$$

$$\neg(\forall R.C) \equiv \exists R.\neg C$$

$$\neg(\exists R.C) \equiv \forall R.\neg C$$

As variables play important role in applications of tableau rules we use them during the proofs with respect to arities of concept and role predicate symbols.

Example 1:

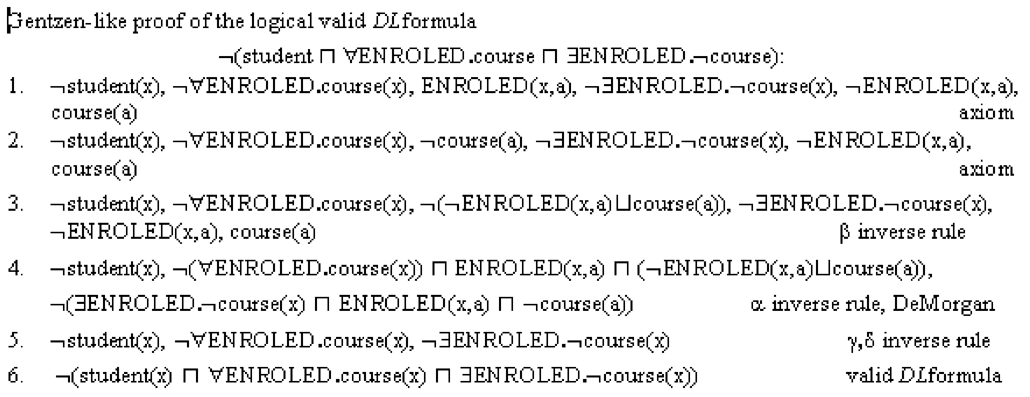

Deciding the consistency of the concept student ⊓ ∀ENROLED.course ⊓ ∃ENROLED.¬course by a direct tableau proof.

student(x), ∀ENROLED.course(x), ∃ENROLED.¬course(x)
α
student(x), ∀ENROLED.course(x), ∃ENROLED.¬course(x), ENROLED(x,a),¬course(a)
δ
student(x), ∀ENROLED.course(x), ¬ENROLED(x,a)⊔course(a), ∃ENROLED.¬course(x), ENROLED(x,a), ¬course(a)
γ
student(x), ∀ENROLED.course(x), ¬ENROLED(x,a),∃ENROLED.¬course(x), ENROLED(x,a), ¬course(a)
student(x), ∀ENROLED.course(x), course(a), ∃ENROLED.¬course(x), ENROLED(x,a), ¬course(a)

We have constructed a closed tableau tree so the root DLformula is inconsistent. Now let us consider a dual tree: commas between DLformulas represent logical disjunctions of negations of the tableau DLformulas, branching of the tree represent logical conjunctions. The label of a leaf now contains a complementary pair of DLformulas and represents a logical valid disjunction. It is possible to consider it represents a logical axiom of a Gentzen-like axiomatic system with dual rules to that of the semantic tableau formal system. On the base of closed tableau tree we can create a dual tree representing a Gentzen-like proof. The leaves of the tree are labeled by Gentzen axioms (disjunctions of DLformulas - their logical validity are ensured by disjunctions of complementary pairs of formulas). Gentzen-like system rules are dual rules to those of the tableau rules

$\alpha, \beta, \gamma, \delta$. In the frame of the Gentzen-like system we can proof (see the following example) the negated DLformula from Gentzen axioms.

Example 2:



Gentzen-like proof of the logical valid *DL*formula

¬(student ⊓ ∀ENROLED.course ⊓ ∃ENROLED.¬course):

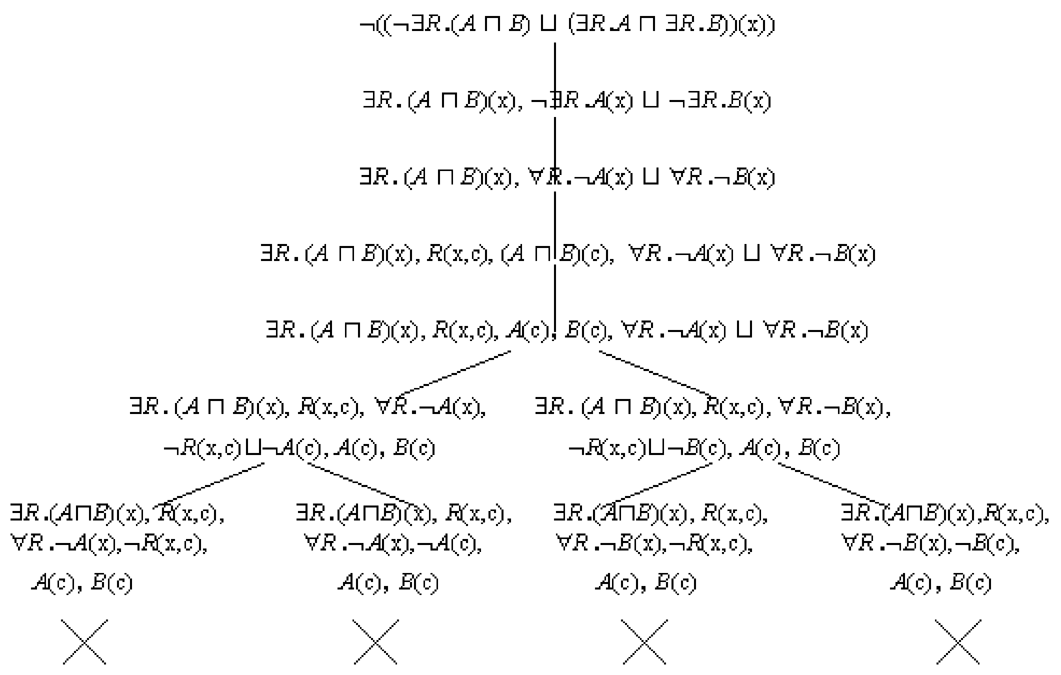
1. ¬student(x), ¬∀ENROLED.course(x), ENROLED(x,a), ¬∃ENROLED.¬course(x), ¬ENROLED(x,a), course(a) — axiom
2. ¬student(x), ¬∀ENROLED.course(x), ¬course(a), ¬∃ENROLED.¬course(x), ¬ENROLED(x,a), course(a) — axiom
3. ¬student(x), ¬∀ENROLED.course(x), ¬(¬ENROLED(x,a)⊔course(a)), ¬∃ENROLED.¬course(x), ¬ENROLED(x,a), course(a) — β inverse rule
4. ¬student(x), ¬(∀ENROLED.course(x)) ⊓ ENROLED(x,a) ⊓ (¬ENROLED(x,a)⊔course(a)), ¬(∃ENROLED.¬course(x) ⊓ ENROLED(x,a) ⊓ ¬course(a)) — α inverse rule, DeMorgan
5. ¬student(x), ¬∀ENROLED.course(x), ¬∃ENROLED.¬course(x) — γ,δ inverse rule
6. ¬(student(x) ⊓ ∀ENROLED.course(x) ⊓ ∃ENROLED.¬course(x)) — valid *DL*formula

Sometimes it may be usefull to add some logical valid formulas into a KB. In the next example we prove some logical valid formulas corresponding to subsumptions that can become a part of an arbitrary KB.

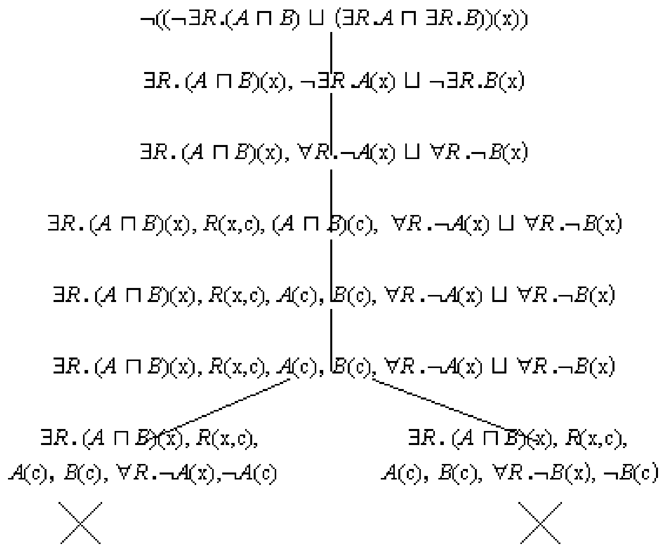
Example 3 Deciding of logical validity of the DLformulas corresponding to following subsumptions:

1. a)$\exists R.(A \sqcap B) \sqsubseteq \exists R.A \sqcap \exists R.B$,
2. b)$\exists R.A \sqcap \exists R.B \sqsubseteq \exists R.(A \sqcap B)$,
3. c)$\forall R.(C \sqcup D) \sqsubseteq \forall R.C \sqcup \forall R.D$,
4. d)$\forall R.C \sqcup \forall R.D \sqsubseteq \forall R.(C \sqcup D)$

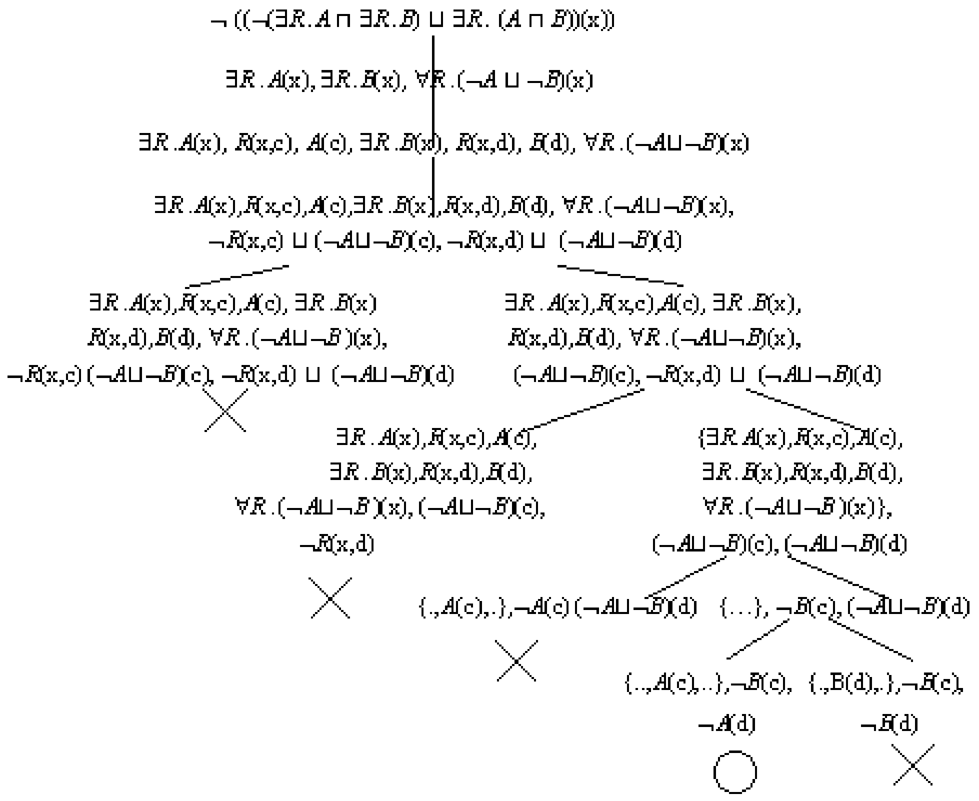
3a-1) Indirect tableau proof of $\exists R.(A \sqcap B) \sqsubseteq \exists R.A \sqcap \exists R.B$ using our tableau algorithm.


$\neg((\neg\exists R.(A \sqcap B) \sqcup (\exists R.A \sqcap \exists R.B))(x))$
$\exists R.(A \sqcap B)(x), \neg\exists R.A(x) \sqcup \neg\exists R.B(x)$
$\exists R.(A \sqcap B)(x), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), (A \sqcap B)(c), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), A(c), B(c), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), \forall R.\neg A(x),$ $\neg R(x,c) \sqcup \neg A(c), A(c), B(c)$
$\exists R.(A \sqcap B)(x), R(x,c), \forall R.\neg B(x),$ $\neg R(x,c) \sqcup \neg B(c), A(c), B(c)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $\forall R.\neg A(x), \neg R(x,c),$ $A(c), B(c)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $\forall R.\neg A(x), \neg A(c),$ $A(c), B(c)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $\forall R.\neg B(x), \neg R(x,c),$ $A(c), B(c)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $\forall R.\neg B(x), \neg B(c),$ $A(c), B(c)$

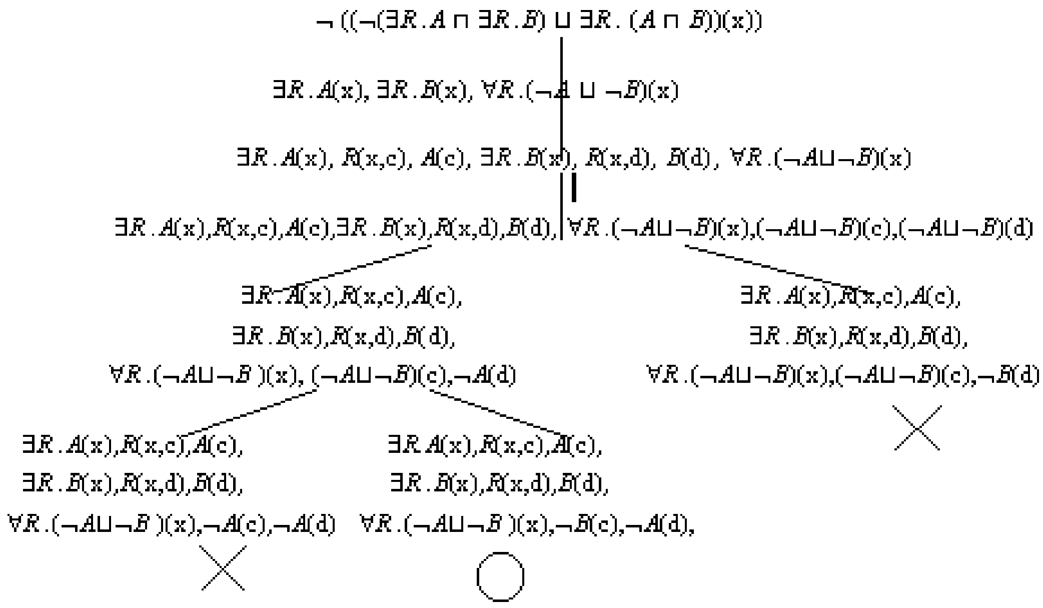
3a-2) Indirect tableau proof of the same subsumption $\exists R.(A \sqcap B) \sqsubseteq \exists R.A \sqcap \exists R.B$ using the tableau algorithm of the [Ba03].


$\neg((\neg\exists R.(A \sqcap B) \sqcup (\exists R.A \sqcap \exists R.B))(x))$
$\exists R.(A \sqcap B)(x), \neg\exists R.A(x) \sqcup \neg\exists R.B(x)$
$\exists R.(A \sqcap B)(x), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), (A \sqcap B)(c), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), A(c), B(c), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c), A(c), B(c), \forall R.\neg A(x) \sqcup \forall R.\neg B(x)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $A(c), B(c), \forall R.\neg A(x), \neg A(c)$
$\exists R.(A \sqcap B)(x), R(x,c),$ $A(c), B(c), \forall R.\neg B(x), \neg B(c)$

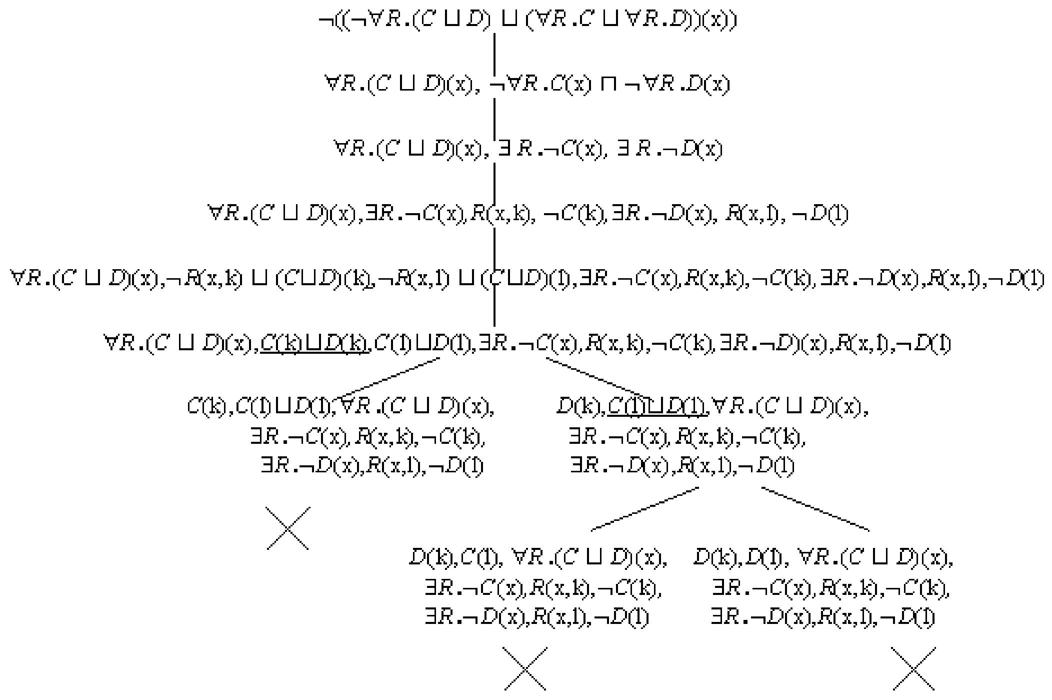
3b-1) Indirect tableau proof of $\exists R.A \sqcap \exists R.B \sqsubseteq \exists R.(A \sqcap B)$ using our tableau algorithm.


¬ ((¬(∃R.A ⊓ ∃R.B) ⊔ ∃R. (A ⊓ B))(x))
∃R.A(x), ∃R.B(x), ∀R.(¬A ⊔ ¬B)(x)
∃R.A(x), R(x,c), A(c), ∃R.B(x), R(x,d), B(d), ∀R.(¬A⊔¬B)(x)
∃R.A(x),R(x,c),A(c),∃R.B(x),R(x,d),B(d), ∀R.(¬A⊔¬B)(x),
¬R(x,c) ⊔ (¬A⊔¬B)(c), ¬R(x,d) ⊔ (¬A⊔¬B)(d)
∃R.A(x),R(x,c),A(c), ∃R.B(x)
R(x,d),B(d), ∀R.(¬A⊔¬B)(x),
¬R(x,c) (¬A⊔¬B)(c), ¬R(x,d) ⊔ (¬A⊔¬B)(d)
∃R.A(x),R(x,c),A(c), ∃R.B(x),
R(x,d),B(d), ∀R.(¬A⊔¬B)(x),
(¬A⊔¬B)(c), ¬R(x,d) ⊔ (¬A⊔¬B)(d)
∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x), (¬A⊔¬B)(c),
¬R(x,d)
{∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x)},
(¬A⊔¬B)(c), (¬A⊔¬B)(d)
{.,A(c),.},¬A(c) (¬A⊔¬B)(d)
{…}, ¬B(c), (¬A⊔¬B)(d)
{..,A(c),..},¬B(c),
¬A(d)
{.,B(d),.},¬B(c),
¬B(d)

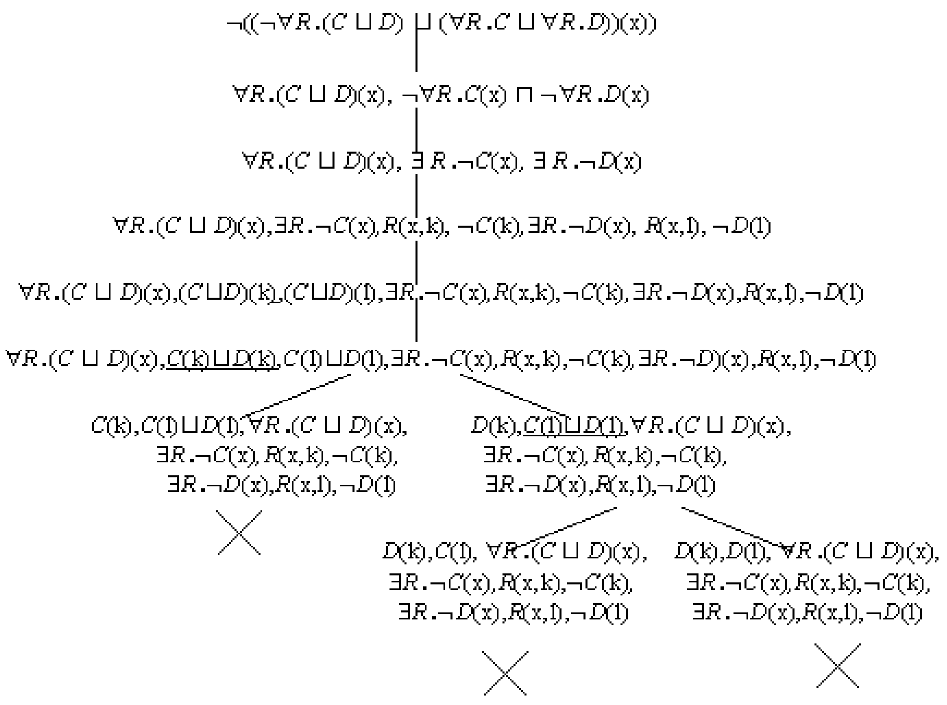
3b-2) Indirect tableau proof of $\exists R.A \sqcap \exists R.B \sqsubseteq \exists R.(A \sqcap B)$ - according to the [Ba03] algorithm:


¬ ((¬(∃R.A ⊓ ∃R.B) ⊔ ∃R. (A ⊓ B))(x))
∃R.A(x), ∃R.B(x), ∀R.(¬A ⊔ ¬B)(x)
∃R.A(x), R(x,c), A(c), ∃R.B(x), R(x,d), B(d), ∀R.(¬A⊔¬B)(x)
∃R.A(x),R(x,c),A(c),∃R.B(x),R(x,d),B(d), ∀R.(¬A⊔¬B)(x),(¬A⊔¬B)(c),(¬A⊔¬B)(d)
∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x), (¬A⊔¬B)(c),¬A(d)
∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x),(¬A⊔¬B)(c),¬B(d)
∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x),¬A(c),¬A(d)
∃R.A(x),R(x,c),A(c),
∃R.B(x),R(x,d),B(d),
∀R.(¬A⊔¬B)(x),¬B(c),¬A(d),

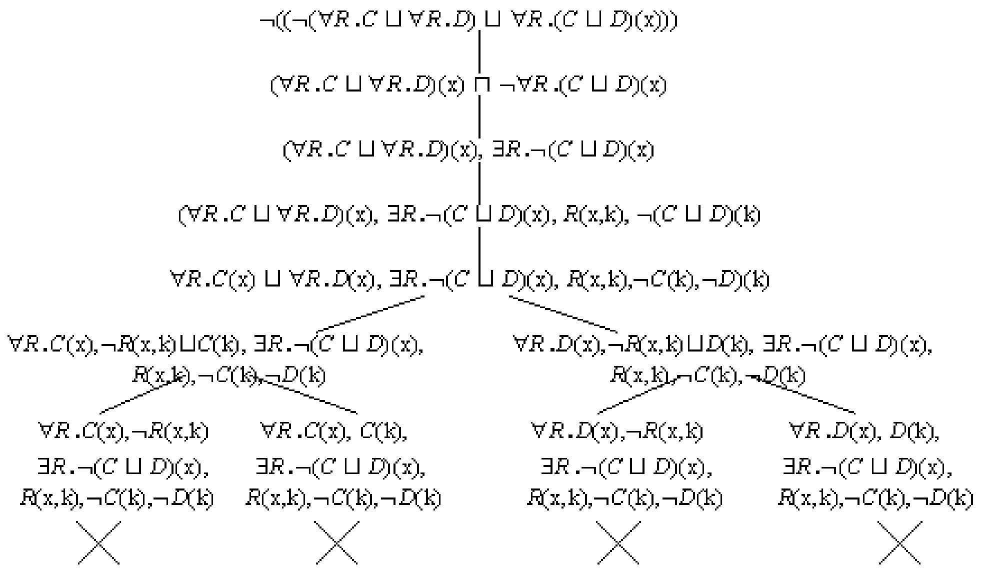
3c-1) Indirect tableau proof of $\forall R.(C \sqcup D) \sqsubseteq \forall R.C \sqcup \forall R.D$ (our tableau algorithm):


$\neg((\neg\forall R.(C \sqcup D) \sqcup (\forall R.C \sqcup \forall R.D))(x))$
$\forall R.(C \sqcup D)(x), \neg\forall R.C(x) \sqcap \neg\forall R.D(x)$
$\forall R.(C \sqcup D)(x), \exists R.\neg C(x), \exists R.\neg D(x)$
$\forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$\forall R.(C \sqcup D)(x), \neg R(x,k) \sqcup (C \sqcup D)(k), \neg R(x,l) \sqcup (C \sqcup D)(l), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$\forall R.(C \sqcup D)(x), \underline{C(k) \sqcup D(k)}, C(l) \sqcup D(l), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D)(x), R(x,l), \neg D(l)$
$C(k), C(l) \sqcup D(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), \underline{C(l) \sqcup D(l)}, \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), C(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), D(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$

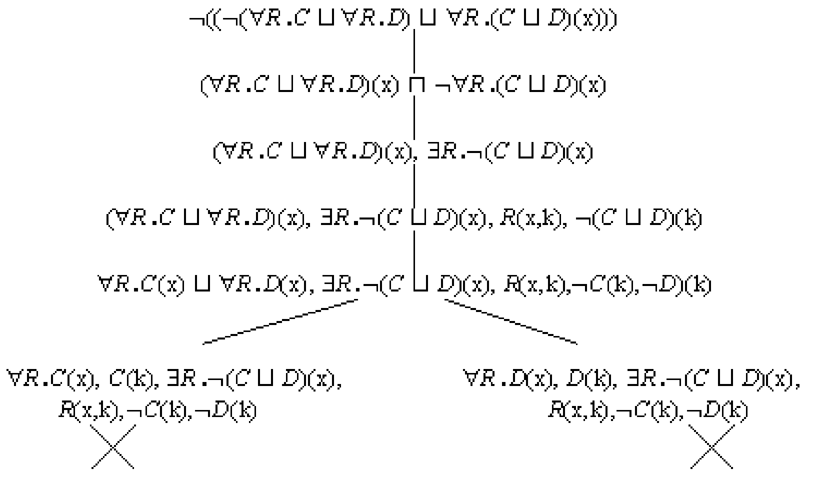
3c-2) Indirect tableau proof of $\forall R.(C \sqcup D) \sqsubseteq \forall R.C \sqcup \forall R.D$ - according to the [Ba03] algorithm:


$\neg((\neg\forall R.(C \sqcup D) \sqcup (\forall R.C \sqcup \forall R.D))(x))$
$\forall R.(C \sqcup D)(x), \neg\forall R.C(x) \sqcap \neg\forall R.D(x)$
$\forall R.(C \sqcup D)(x), \exists R.\neg C(x), \exists R.\neg D(x)$
$\forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$\forall R.(C \sqcup D)(x), (C \sqcup D)(k), (C \sqcup D)(l), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$\forall R.(C \sqcup D)(x), \underline{C(k) \sqcup D(k)}, C(l) \sqcup D(l), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D)(x), R(x,l), \neg D(l)$
$C(k), C(l) \sqcup D(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), \underline{C(l) \sqcup D(l)}, \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), C(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$
$D(k), D(l), \forall R.(C \sqcup D)(x), \exists R.\neg C(x), R(x,k), \neg C(k), \exists R.\neg D(x), R(x,l), \neg D(l)$

3d-1) Indirect tableau proof of $\forall R.C \sqcup \forall R.D \sqsubseteq \forall R.(C \sqcup D)$:


¬((¬(∀R.C ⊔ ∀R.D) ⊔ ∀R.(C ⊔ D)(x)))
(∀R.C ⊔ ∀R.D)(x) ⊓ ¬∀R.(C ⊔ D)(x)
(∀R.C ⊔ ∀R.D)(x), ∃R.¬(C ⊔ D)(x)
(∀R.C ⊔ ∀R.D)(x), ∃R.¬(C ⊔ D)(x), R(x,k), ¬(C ⊔ D)(k)
∀R.C(x) ⊔ ∀R.D(x), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D)(k)
∀R.C(x),¬R(x,k)⊔C(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.D(x),¬R(x,k)⊔D(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.C(x),¬R(x,k) ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.C(x), C(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.D(x),¬R(x,k) ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.D(x), D(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)

3d-2) Indirect tableau proof of the $\forall R.C \sqcup \forall R.D \sqsubseteq \forall R.(C \sqcup D)$ - according to the [Ba03] algorithm:

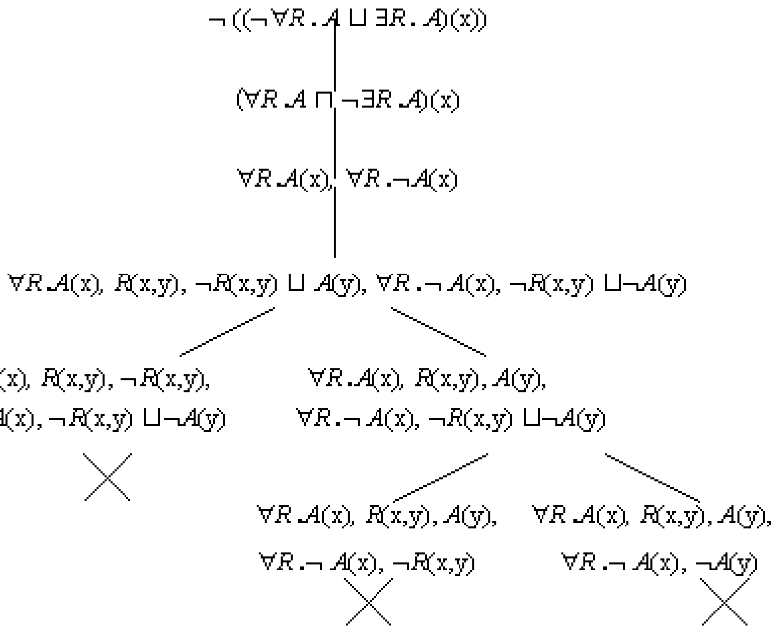

¬((¬(∀R.C ⊔ ∀R.D) ⊔ ∀R.(C ⊔ D)(x)))
(∀R.C ⊔ ∀R.D)(x) ⊓ ¬∀R.(C ⊔ D)(x)
(∀R.C ⊔ ∀R.D)(x), ∃R.¬(C ⊔ D)(x)
(∀R.C ⊔ ∀R.D)(x), ∃R.¬(C ⊔ D)(x), R(x,k), ¬(C ⊔ D)(k)
∀R.C(x) ⊔ ∀R.D(x), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D)(k)
∀R.C(x), C(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)
∀R.D(x), D(k), ∃R.¬(C ⊔ D)(x), R(x,k),¬C(k),¬D(k)

As the tableau tree in the example 3a) (3c), 3d)) closes the negation of the DLformula corresponding to the subsumption a)(c), d)) must be inconsistent and so the DLformula corresponding to a) (c), d)) must be valid. In the case of b) we can't find a closed tableau tree, one of the branches stay open and it is obvious that creating of further instances by the use of the γ , δ rules can't lead to a complementary pair of DLformulas (see [Lu05]). Nevertheless we can obtain in all the cases 3a)- 3d) the same results with the rules of the [Ba03]. A question arises why the author of this paper recommends the γ -rule of the

def.10 instead of that one of the [Ba03]. The answer follows from example of the indirect tableau proof of logical validity of subsumption $\forall R.A \sqsubseteq \exists R.A$ (example 4). If we had to our disposal only the tableau rules of the [Lu03] we wouldn't be able to proof the valid subsumption $\forall R.A \sqsubseteq \exists R.A$.
It is because DLformulas $\forall R.A(x)$, $\forall R.\neg A(x)$ at the 3rd row doesn't represent a complementary pair of formulas and as there is no one rule to apply at the level of the tree, the algorithm stops. Our γ -rule adds to the list of DLformulas two new formulas $R(x,y)$, $\neg R(x,y) \sqcup C(y)$ expressing correctly the semantics of the rule $\forall R.C$. On the other hand the $\forall$ -rule of the [Ba03] adds to the list only the logical consequence $C(y)$ of these two DLformulas and moreover it assumes $R(x,y)$ be present in the list before applying the $\forall$ -rule.

Example 4:

4-1) Indirect tableau proof of logical validity of the subsumption $\forall R.A \sqsubseteq \exists R.A$:



$\neg((\neg\forall R.A \sqcup \exists R.A)(x))$
$(\forall R.A \sqcap \neg\exists R.A)(x)$
$\forall R.A(x), \forall R.\neg A(x)$
$\forall R.A(x), R(x,y), \neg R(x,y) \sqcup A(y), \forall R.\neg A(x), \neg R(x,y) \sqcup \neg A(y)$
$\forall R.A(x), R(x,y), \neg R(x,y), \forall R.\neg A(x), \neg R(x,y) \sqcup \neg A(y)$
$\forall R.A(x), R(x,y), A(y), \forall R.\neg A(x), \neg R(x,y) \sqcup \neg A(y)$
$\forall R.A(x), R(x,y), A(y), \forall R.\neg A(x), \neg R(x,y)$
$\forall R.A(x), R(x,y), A(y), \forall R.\neg A(x), \neg A(y)$

4-2) Indirect tableau proof of logical validity of $\forall R.A \sqsubseteq \exists R.A$ using the algorithm of [Ba03]

$$\neg((\neg\forall R.A \sqcup \exists R.A)(x))$$

$$|$$

$$(\forall R.A \sqcap \neg\exists R.A)(x)$$

$$|$$

$$\forall R.A(x),\ \forall R.\neg A(x)$$

$$\bigcirc$$

In our last example we deduce some logical consequences of a part of the KB S ([Bu93]), example 3) in the Gentzen-like formal system using dual rules to our semantic tableau formal system.

Example 5:

Gentzen-like deduction on the knowledge base Σ [2] rewritten into a set **S** of *DL* formulas:

TBox :

P1 : $\exists$TEACHES.course $\sqsubseteq$ (student $\sqcap$ $\exists$ABSOLVEN.bc) $\sqcup$ pedagog

$\neg\exists$TEACHES.course $\sqcup$ (student $\sqcap$ $\exists$ABSOLVENT.bc) $\sqcup$ pedagog

P2: pedagog $\sqsubseteq$ $\exists$ABSOLVENT.mgr

$\neg$pedagog $\sqcup$ $\exists$ABSOLVENT.mgr

P3: $\exists$ABSOLVENT.mgr $\sqsubseteq$ $\exists$ABSOLVENT.bc

$\neg\exists$ABSOLVENT.mgr $\sqcup$ $\exists$ABSOLVENT.bc

P4: mgr $\sqcap$ bc $\sqsubseteq$ $\perp$

$\neg$(mgr $\sqcap$ bc)

ABox :

TEACHES (John, dl01)

$\exists^{\leq 1}$ ABSOLVENT(John)

course (dl01)

Gentzen-like proof:

1. P1 $\vdash$ $\neg\exists$ TEACHES . course, (student $\sqcap$ $\exists$ ABSOLVENT. bc), pedagog
2. P2 $\vdash$ $\neg$pedagog, $\exists$ ABSOLVENT. mgr
3. P3 $\vdash$ $\neg\exists$ ABSOLVENT. mgr, $\exists$ ABSOLVENT . bc
4. P2, P3 $\vdash$ $\neg$pedagog, $\exists$ ABSOLVENT. mgr $\sqcap$ $\neg\exists$ ABSOLVENT. mgr,
   $\exists$ ABSOLVENT . bc   dual α-rule on 2,3
5. P2, P3 $\vdash$ $\neg$pedagog, $\exists$ ABSOLVENT. bc   contraction (cut)
6. P1, P2, P3 $\vdash$ $\neg\exists$ TEACHES . course, (student $\sqcap$ $\exists$ ABSOLVENT. bc), pedagog $\sqcap$ $\neg$pedagog,
   $\exists$ ABSOLVENT. bc α-rule on 1,5
7. P1, P2, P3 $\vdash$ $\neg\exists$ TEACHES . course, (student $\sqcap$ $\exists$ ABSOLVENT. bc),
   $\exists$ ABSOLVENT. bc   contraction (cut)
8. P1, P2, P3 $\vdash$ $\neg\exists$ TEACHES . course $\sqcup$ (student $\sqcap$ $\exists$ ABSOLVENT. bc) $\sqcup$
   $\exists$ ABSOLVENT. bc   2x β-rule

DL formula 8 rewritten:

$\exists$ TEACHES . course $\sqsubseteq$ (student $\sqcap$ $\exists$ ABSOLVENT. bc) $\sqcup$ $\exists$ ABSOLVENT. bc

References

[Ba03] Baader et col. (eds.): The Description Logic Handbook - Theory, Interpretation and Applications. Cambridge Univ. Press, 2003.

[Bu93] Buchheit, M., Donini, F., Schaerf, A.: Decidable Reasoning in Terminological Knowledge Representation Systems. Journal of Artificial Intelligence Research 1, 1993, 109 - 138.

[Lu03] Lukasová, A.: Formal Logics in Artificial Intelligence (in Czech). Computer Press, 2003, Printed in Czech Republic.

[Lu05] Lukasová, A.: Reasoning in description logic by semantic tableau binary tree. Research Report No. 63, IFM , University of Ostrava, 2005