Available online at www.sciencedirect.com

ScienceDirect

Journal of Applied Logic 6 (2008) 343–360

JOURNAL OF APPLIED LOGIC

www.elsevier.com/locate/jal

Inconsistency-tolerant description logic. Part II: A tableau algorithm for $\mathcal{CALC}^{\mathsf{C}}$

S.P. Odintsov [a,*], H. Wansing [b]

[a] *Sobolev Institute of Mathematics, Novosibirsk, Russia*
[b] *Institute of Philosophy, Dresden University of Technology, Germany*

Available online 20 June 2007

Abstract

In the first part of this paper, we motivated and defined three systems of constructive and inconsistency-tolerant description logic. The variety of arising systems is conditioned by the variety of approaches to defining modalities in the constructive setting. We also presented sound and complete tableau calculi for the logics under consideration. Whereas these calculi were not meant to give rise to tableau algorithms, in the present second part of the paper, after providing some motivation and recalling the main definitions, we adapt methods developed by R. Dyckhoff and by I. Horrocks and U. Sattler in order to define a tableau algorithm for our basic four-valued constructive description logic $\mathcal{CALC}^{\mathsf{C}}$. Notice that among the three logics defined in the first part of the paper, $\mathcal{CALC}^{\mathsf{C}}$ is the only logic which lends itself to applications, because for the other logics it is unknown whether they are elementarily decidable. The presented algorithm for $\mathcal{CALC}^{\mathsf{C}}$ is the first example of an elementary decision procedure for a constructive description logic.

Keywords: Description logic; Tableau algorithms; Constructive logic; Paraconsistent logic

1. Introduction

If knowledge representation is understood as the faithful representation of facts about the world, then it would seem that the representation ought to be consistent. As a matter of fact, the information we have about the world, however, is often inconsistent. The merging of data, for example, may lead to contradictory pieces of information. If knowledge representation amounts to *information representation*, inconsistency-tolerant reasoning mechanisms may be used in order to avoid trivialization. In the first part of the present paper, we motivated three systems of inconsistency-tolerant constructive description logic: $\mathcal{CALC}^{\mathsf{C}}$ (a constructive version of the basic description logic $\mathcal{ALC}$ with a semantics induced by a translation into classical predicate logic), $\mathcal{CALC}^{\mathsf{N4}}$ (a version of $\mathcal{ALC}$ with a semantics induced by a translation into the constructive predicate logic QN4), and $\mathcal{CALC}^{\mathsf{N4d}}$ (a version of $\mathcal{ALC}$ with a semantics induced by another translation into QN4 guaranteeing the duality of existential and universal role restrictions), see also [13]. We shall not rehearse here the background information about description logic and the motivational remarks of Part I. However, in order to keep Part II reasonably self-contained, we shall briefly address motivational issues and repeat the

[*] Corresponding author.
E-mail addresses: odintsov@math.nsc.ru (S.P. Odintsov), Heinrich.Wansing@tu-dresden.de (H. Wansing).

1570-8683/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.jal.2007.06.001

semantical definitions of $\mathcal{CALC}^{\mathsf{C}}$. In [12] we showed that $\mathcal{CALC}^{\mathsf{N4d}}$ is decidable, but this was done by a translation of $\mathcal{CALC}^{\mathsf{N4}d}$ into Fischer Servi's intuitionistic modal logic. It is known that Fischer Servi logic is decidable [7,18], but it is still unknown whether it is elementarily decidable. The complexity of the decidability algorithm presented in [18], for instance, is estimated by the Ackerman function. So it is still unknown whether the logic $\mathcal{CALC}^{\mathsf{N4d}}$ can be used for applications. The situation with the logic $\mathcal{CALC}^{\mathsf{C}}$ is different. In the present paper we adapt methods from [5] and [8] to prove decidability of $\mathcal{CALC}^{\mathsf{C}}$ by presenting an effective tableau algorithm for this basic system of constructive description logic. We show that the algorithm has the same complexity as the tableau algorithm for the description logic with transitive roles presented in [8]. Notice that the main technical problem is connected not with the paraconsistent nature of the negation operator, but with the constructive interpretation of the implication connective $\supseteq$ from the language of $\mathcal{CALC}^{\mathsf{C}}$. The connective $\supseteq$ is interpreted with the help of an additional transitive role relation $\preccurlyeq$. Moreover, in the construction of a tableau, some concept descriptions must be transfered along edges marked with $\preccurlyeq$. This situation is similar to that of [8], where tableaux for the description logic with transitive roles was defined, and as in [8] it leads to infinite tableau branches. This problem was successively solved in [8] via a loop-detecting mechanism, but in our case there is also an interplay of $\preccurlyeq$ with other role relations, which makes problems for creating loops in the tableaux. For this reason we avoid infinite branches as it was done in [5] by introducing a special system of rules treating antecedents of the constructive implication.

2. Motivation for paraconsistent constructive description logics

Recently, Valeria de Paiva [14] raised the question: "Why should we worry about describing systems of *constructive* description logics?" and listed a number of quite general reasons why the development of constructive description logics is useful. According to de Paiva, "being able to construct and prove results about the constructive or intuitionistic based version of your logic, is a kind of *casting out nines* test for that logic. If you cannot do it, there is probably something wrong with your system". Moreover, she argues (i) that the interest in decidable predicates in knowledge representation "should lead to calculi that are basically constructive", (ii) that "a constructive logic has some intrinsic flavour of partiality", which might be useful for representing the meaning of natural language sentences, and (iii) that a Curry–Howard isomorphism is easier to obtain for constructive logics, which might then give rise to a useful type theory based on a constructive description logic.

The best-known system of constructive logic is intuitionistic logic. One attractive feature of intuitionistic logic is that its relational possible-worlds semantics admits of an 'informational' interpretation according to which the possible worlds are information states and the assumed accessibility relation between worlds is a relation of possible expansion of information states. With this understanding, the persistence (or heredity) condition for atomic formulas makes sense, because it amounts to assuming that atomic information is retained by moving from an information state to its possible expansions. This semantics, however, puts a single-sided emphasis on the notion of verification. Whereas an atomic formula may be *verified* directly at a state, there is no analogous possibility of *falsifying* an atomic formula on the spot. An information state may support the truth of an atom, but it may not by itself support the falsity of an atom. The latter concept is not definable by means of intuitionistic negation, because the intuitionistic negation of an atom is true at a state a iff the atom is verified at every possible expansion of a. It has therefore been suggested to supplement the intuitionistic support-of-truth relation by an independent support-of-falsity relation and to introduce a negation operation $\sim$ that allows one to change from support of truth to support of falsity, such that a state a supports the truth of a formula φ iff a supports the falsity of $\sim\varphi$. This strong, constructive negation has been investigated by David Nelson, and his system N4 of constructive logic with strong negation forms the basic system of constructive four-valued logic. In the relational semantics of N4, not only support of truth but also support of falsity is inherited from an information state to its possible expansions. Moreover, the support-of-falsity conditions for compound formulas are explicitly spelled out. The logic N4 is an inconsistency-tolerant system of *paraconsistent* logic. The rationale for this is that information can come from different sources, so that it may happen that an information state is such that it supports the truth and also supports the falsity of a certain proposition. This approach can also be extended to substructural subsystems of N4, see for example, [10] and [20]. A completeness proof for quantified N4, the system QN4 referred to in Section 1, is presented in Part I of this paper [12].

The observation that inconsistency-tolerant description logics have useful applications is not new. Four-valued paraconsistent description logics have been suggested not only in [12] but also by Patel-Schneider [15], Meghini and Straccia [11] and Straccia [19]. Straccia, for example, emphasizes that since the *content representation* of a collection of documents cannot be considered as a consistent set of assertions an adequate D[ocument]R[etrieval] model must be capable of tolerating inconsistencies without loosing its deductive capabilities. [19, p. 344]

Patel-Schneider's and Straccia's paraconsistent description logics are based on Belnap's useful four-valued logic, also known as *first degree entailment* (*FDE*), see [1–4], [16, Chap. 8], [17]. A central reasoning task in Description Logic is subsumption checking, finding out whether a concept *subsumes* another concept. On a classical understanding, a concept (description) C subsumes a concept (description) D iff the interpretation of D is a subset of the interpretation of C. There are two important things to note concerning the notion of subsumption in description logics based on *FDE*. First, in *FDE* the positive extension of a concept is distinguished from its negative extension (or anti-extension), and it may happen that an individual a belongs to both the extension and the anti-extension of a given concept description C or that a neither belongs to the extension nor to the anti-extension of C. Subsumption is defined as the set-inclusion of positive extensions only, and the four-valued subsumption relations of Patel-Schneider and Straccia form a proper subset of the classical two-valued subsumption relationships. The extension of a negated concept $\sim C$ coincides with the anti-extension of C. Therefore, the set-inclusion of anti-extensions may be expressed as the subsumption of the negated concepts. A contradictory concept description $(D \sqcap \sim D)$, for example, is not subsumed by every concept description. Second, *FDE* has no theorems, and there is no implication satisfying the Deduction Theorem. An implication $(C \supseteq D)$ is defined as a material implication $(\sim C \sqcup D)$, and modus ponens for material implication (also knows as Disjunctive Syllogism (*DS*)) fails:

$$\big(C \sqcap (\sim C \sqcup D)\big)(a) \not\models D(a).$$

The failure of the *DS* in general is unproblematic for some authors. As Priest [16, p. 152] explains:

It may be perfectly legitimate to use it ... There are a number of ways of spelling this idea out in detail, but at the root of all of them is the observation that when the *DS* fails, it does so because the premise ... involved is a truth-value glut. If the situation about which we are reasoning is consistent ... the *DS* cannot lead us from truth to untruth. So it is legitimate to use it. This fact will underwrite its use in most situations we come across, since consistency is, arguably, the norm.

Whereas the failure of the following restricted form of *DS*:

$$\big((C \sqcap \sim C) \sqcap \big(\sim (C \sqcap \sim C) \sqcup D\big)\big)(a) \not\models D(a),$$

where the minor premise of modus ponens is contradictory, may be tolerable, for a genuine implication $\supseteq$ one would like to have:

$$\big((C \sqcap \sim C) \sqcap \big((C \sqcap \sim C) \supseteq D\big)\big)(a) \models D(a).$$

Concept descriptions of the form $((C \sqcap \sim C) \supseteq D)$ may naturally arise in applications. The knowledge base Σ_1 presented in [11] contains information about sending orders for selling cars and sending invoices. Its extension $\Sigma_1' = \Sigma_1 \cup \{\texttt{PrivateVendor(v}_1\texttt{), Order(o}_3\texttt{)}\}$ entails that vendor $\texttt{v}_1$ is both a reseller and not a reseller of cars: $\Sigma_1' \models (\texttt{Reseller} \sqcap \sim \texttt{Reseller})(\texttt{v}_1)$. It could be wanted to mark certain, but not all, individuals for the possibility of obtaining the information that they satisfy $(\texttt{Reseller} \sqcap \sim \texttt{Reseller})$. It might be of interest, for example, to extend Σ_1 by the assertion $((\texttt{Reseller} \sqcap \sim \texttt{Reseller}) \supseteq \texttt{Suspicious})(\texttt{v}_1)$. There are thus reasons for refraining from representing an implication $(C \supseteq D)$ as a material implication $(\sim C \sqcup D)$ in the four-valued setting. The absence of a genuine implication satisfying the Deduction Theorem may be seen as a serious drawback of the paraconsistent description logics of Patel-Schneider and Meghini and Straccia.

The systems $\mathcal{CALC}^{\mathsf{C}}$, $\mathcal{CALC}^{\mathsf{N4}}$ and $\mathcal{CALC}^{\mathsf{N4d}}$ of constructive four-valued description logic share with Patel-Schneider's and Straccia's four-valued systems the non-classicality of the subsumption relation. However, in $\mathcal{CALC}^{\mathsf{C}}$, $\mathcal{CALC}^{\mathsf{N4}}$ and $\mathcal{CALC}^{\mathsf{N4d}}$ intuitionistic implication, the smallest implication satisfying the Deduction Theorem (for single-conclusion derivations from sets of premises), is a primitive concept description forming operation. In these systems, subsumption of D by C may be defined as valid implication $(\models (D \supseteq C))$, which may be understood as valid set inclusion for informationally accessible individuals, cf. Section 3. In the purely implicational language of $\supseteq$, the difference to the classical notion of subsumption can be neatly identified as the failure of Peirce's Law:

$$\not\models \big(((C \supseteq D) \supseteq C) \supseteq C\big),$$

and in [12] it is argued that this loss is quite acceptable.

Moreover, the inclusion of intuitionistic implication also leads to some welcome properties. As we shall recall for $\mathcal{CALC}^{\mathsf{C}}$ in Section 3, we have the disjunction property and the constructible falsity property.

Constructive implication not only has good proof theoretical properties, it also allows one to take into account the positive dynamics of data bases. Assume that the data base may be supplemented in due course. Different states of a data base are denoted as $t, q, t', q', \ldots$ and will be called later information states. Denote by $t \preccurlyeq t'$ the fact that the state t' follows after t. If we have some unit of information (positive or negative) at a state t, say $C(a)$ and $\sim D(b)$, i.e. that a belongs to the extension of C and b belongs to the anti-extension of D, this information will be preserved later on:

$$t \preccurlyeq t', \quad t \models C(a), \quad t \models \sim D(b) \Rightarrow t' \models C(a), \quad t' \models \sim D(b).$$

The implication is interpreted so that if $t \models (C \supseteq D)(a)$, then for all t',

$$t \preccurlyeq t', \quad t' \models C(a) \Rightarrow t' \models D(a).$$

In this way, the validity of $(C \supseteq D)(a)$ at a state t is a sort of rule, which adds later on a to the extension of D as soon as it was added to the extension of C.

Note that the positive dynamics of a data base it is all what can be taken into account at the level of an ordinary monotonic deductive system. If we want to take into account the negative dynamics, i.e. the possibility to delete information from a data base or to replace one fragment of information by another one, we have to deal with non-monotonic logics and belief revision theories, which are higher order superstructures over monotonic deductive systems.

3. Syntax and semantics of $\mathcal{CALC}^{\mathsf{C}}$

We first recall from [12] the syntax and semantics of $\mathcal{CALC}^{\mathsf{C}}$. The language of $\mathcal{CALC}^{\mathsf{C}}$ consists of a non-empty set N_C of concept names, a non-empty set N_r of role names and a set of concept constructors used to build complex concept descriptions. The syntax is defined as follows:

concept names:	$A \in N_C$
role names:	$r \in N_r$
concept descriptions:	$C \in Con(N_C, N_r)$

$C ::= A \mid\, \sim C \mid (C \sqcap D) \mid (C \sqcup D) \mid (C \supseteq D) \mid \forall r.C \mid \exists r.C.$

We shall sometimes omit outermost brackets.

The semantics for $\mathcal{CALC}^{\mathsf{C}}$ uses interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \preccurlyeq, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the individual domain) and $\preccurlyeq \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ is a reflexive and transitive relation of *informational accessibility*. We may think of an element of $\Delta^{\mathcal{I}}$ as an *individual at a state*, where the state (understood as an information state) is left implicit, i.e. x is an abbreviation for the pair (a, t), where a is an individual and t is an information state. If x is an individual at a state and y is an individual at a state, then y is *informationally accessible* from x (in symbols $x \preccurlyeq y$), if y is the same as x, however, at a state that is a possible expansion of the state of x. In other words, if $x \preccurlyeq y$, then $x = (a, t)$ and $y = (a, t')$, moreover $t \preccurlyeq t'$. The interpretation function $\cdot^{\mathcal{I}}$ maps every $A \in N_C$ and every $\sim A$ with $A \in N_C$ to a subset of $\Delta^{\mathcal{I}}$, and it maps every role name $r \in N_r$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to arbitrary concept descriptions as stated in Table 1. Moreover, we require

(P1) for all $x, y \in \Delta^{\mathcal{I}}$, $A \in N_C$: if $x \preccurlyeq y$ and $x \in A^{\mathcal{I}}$, then $y \in A^{\mathcal{I}}$;
(P2) for all $x, y \in \Delta^{\mathcal{I}}$, $A \in N_C$: if $x \preccurlyeq y$ and $x \in {\sim}A^{\mathcal{I}}$, then $y \in {\sim}A^{\mathcal{I}}$;
(P3) for all $r \in N_r$: $\preccurlyeq^{-1} r^{\mathcal{I}} \subseteq r^{\mathcal{I}} \preccurlyeq^{-1}$.

The relation $x \in C^{\mathcal{I}}$ should be understood as follows. If $x = (a, t)$, then $C(a)$ holds at a state t, $t \models C(a)$. The reason that we do not use explicit denotations for states is as follows. Our semantics allows to track changes of

Table 1
The semantics of concept descriptions in $\mathcal{CALC}^{\mathsf{C}}$

Constructor	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\sim A$	$\sim A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\sim\sim C$	$C^{\mathcal{I}}$
$(C \sqcap D)$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\sim (C \sqcap D)$	$\sim C^{\mathcal{I}} \cup \sim D^{\mathcal{I}}$
$(C \sqcup D)$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\sim (C \sqcup D)$	$\sim C^{\mathcal{I}} \cap \sim D^{\mathcal{I}}$
$(C \sqsupseteq D)$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x \preccurlyeq y \Rightarrow (y \in C^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}}))\}$
$\sim (C \sqsupseteq D)$	$C^{\mathcal{I}} \cap \sim D^{\mathcal{I}}$
$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \forall z\ ((x \preccurlyeq y\ \&\ y r^{\mathcal{I}} z) \Rightarrow z \in C^{\mathcal{I}})\}$[a]
$\sim \forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x r^{\mathcal{I}} y\ \&\ y \in \sim C^{\mathcal{I}})\}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x r^{\mathcal{I}} y\ \&\ y \in C^{\mathcal{I}})\}$
$\sim \exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \forall z\ ((x \preccurlyeq y\ \&\ y r^{\mathcal{I}} z) \Rightarrow z \in \sim C^{\mathcal{I}})\}$

[a] This and the last line are modified as compared to [12]. In [12] the interpretation of $\forall r.C$ was defined as $\{x \in \Delta^{\mathcal{I}} \mid \forall y (x r^{\mathcal{I}} y \Rightarrow y \in C^{\mathcal{I}})\}$ and the interpretation of $\sim \exists r.C$ in a dual way as $\{x \in \Delta^{\mathcal{I}} \mid \forall y (x r^{\mathcal{I}} y \Rightarrow y \in \sim C^{\mathcal{I}})\}$. To guarantee persistence of arbitrary concept descriptions we imposed an additional condition on role interpretations:

$$\text{for all } r \in N_r\colon \preccurlyeq r^{\mathcal{I}} \subseteq r^{\mathcal{I}} \preccurlyeq.$$

It is well known from intuitionistic modal logic that these approaches are equivalent (see, e.g., [18]). Note also that in [12] we wrote $(C \sqsubseteq D)$ instead of $(C \sqsupseteq D)$.

$$\begin{tikzcd}
z \arrow[r, "r"] & t \\
x \arrow[u, "\preccurlyeq"] \arrow[r, "r"'] & y \arrow[u, "\preccurlyeq"']
\end{tikzcd}$$

Fig. 1.

information on individuals, but it does not allow to distinguish particular states of a data base. If $x = (a, t)$, $y = (b, t')$, and $x \preccurlyeq y$, then $a = b$ and $t \preccurlyeq t'$. But if x and y are incomparable wrt information accessibility, then we are sure that $a \neq b$, but we can say nothing on the interrelation between t and t'. Introducing in the semantics explicit information states is very expensive from the complexity point of view. For example, Fischer Servi's logic [6,18] is based on the semantics with explicit states, but as was mentioned in the introduction, it is still unknown whether this logic is elementarily decidable.

Conditions (P1) and (P2) mean that if some positive or negative fact is known about an individual at a state t, then it holds true at all later states. In other words, the interpretations of concept names and of negations of concept names are persistent wrt information accessibility.

Condition (P3) has the following explanation (see Fig. 1).

Assume that individuals x and y from $\Delta^{\mathcal{I}}$ are connected by a role relation $r^{\mathcal{I}}$. We obtain new information $\forall r.C$ about x, which is transformed in this way to z, $x \preccurlyeq z$, $z \in (\forall r.C)^{\mathcal{I}}$, $x \notin (\forall r.C)^{\mathcal{I}}$. Taking into account that z is the same individual as x but in the new state, the relation $x r^{\mathcal{I}} y$ is inherited by z, $z r^{\mathcal{I}} y$. In this way, concept C should be transferred through the r-edge to y. If $y \notin C^{\mathcal{I}}$, there arise a new element t with $t \in C^{\mathcal{I}}$ and $y \preccurlyeq t$. Note that $z \preccurlyeq^{-1} r^{\mathcal{I}} y$ and t proves that $z r^{\mathcal{I}} \preccurlyeq^{-1} y$.

Condition (P3) is used to prove persistence of arbitrary concept descriptions wrt $\preccurlyeq$.

Proposition 1 *(Persistence). Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \preccurlyeq, \cdot^{\mathcal{I}})$ be an interpretation. Then for every concept description C and $x, y \in \Delta^{\mathcal{I}}$, if $x \preccurlyeq y$ and $x \in C^{\mathcal{I}}$, then $y \in C^{\mathcal{I}}$.*

The syntax of description logics is extended by a set of individual names N_I. This extension allows the formulation of simple assertions that may be collected in so-called ABoxes.

Definition 2. An ABox is a finite set of expressions of the form $C(a)$ or $r(a,b)$, where $a, b \in N_I$, $r \in N_r$, and $C \in Con(N_C, N_r)$. An expression $C(a)$ is called a concept assertion, and an expression $r(a,b)$ is called a role assertion. Concept assertions and role assertions are also called ABox statements.

The interpretation function $\cdot^{\mathcal{I}}$ may be extended to apply also to individual names such that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Such an interpretation is a model of an ABox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) iff for every $C(a) \in \mathcal{A}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and for every $r(a,b) \in \mathcal{A}$, $a^{\mathcal{I}} r^{\mathcal{I}} b^{\mathcal{I}}$.

Definition 3. $C \in Con(N_C, N_r)$ is said to be $\mathcal{CALC}^{\mathsf{C}}$-valid ($\models_{\mathcal{CALC}^{\mathsf{C}}} C$) iff for every interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \preccurlyeq, \cdot^{\mathcal{I}} \rangle$, $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. C is subsumed by D ($C \sqsubseteq D$) iff $\models_{\mathcal{CALC}^{\mathsf{C}}} C \supseteq D$. C and D are positively equivalent iff for every interpretation $\mathcal{I}$, $(C \supseteq D)^{\mathcal{I}} = (D \supseteq C)^{\mathcal{I}} = \Delta^{\mathcal{I}}$. C and D are negatively equivalent iff for every interpretation $\mathcal{I}$, $(\sim C \supseteq \sim D)^{\mathcal{I}} = (\sim D \supseteq \sim C)^{\mathcal{I}} = \Delta^{\mathcal{I}}$. C and D are fully equivalent iff C and D are both positively and negatively equivalent.

We shall use the following definitions:

$$\begin{array}{lll}(C =^{+} D) & \text{iff}_{def} & (C \sqsubseteq D \text{ and } D \sqsubseteq C),\\ (C =^{-} D) & \text{iff}_{def} & ({\sim}C \sqsubseteq {\sim}D \text{ and } {\sim}D \sqsubseteq {\sim}C),\\ (C \equiv D) & \text{iff}_{def} & (C =^{+} D \text{ and } C =^{-} D).\end{array}$$

Note that in this paraconsistent setting, *every* concept description is satisfiable. $C^{\mathcal{I}} \neq \varnothing$ means that there is an $x \in \Delta^{\mathcal{I}}$ such that if $a^{\mathcal{I}} = x$, the information that $C(a)$ is available. But this availability is always possible, because the relevant information may be supplied by some source of information.

Definition 4. A TBox is a finite set of expressions of the following shape: $(C \sqsubseteq D)$, $(C =^{+} D)$, $(C =^{-} D)$, or $(C \equiv D)$. The elements of a TBox are called TBox statements. An interpretation $\mathcal{I}$ is a model of $(C \sqsubseteq D)$ iff $(C \supseteq D)^{\mathcal{I}} = \Delta^{\mathcal{I}}$; $\mathcal{I}$ is a model of $(C =^{+} D)$ (respectively $(C =^{-} D)$) iff $\mathcal{I}$ is a model of $(C \sqsubseteq D)$ and $(D \sqsubseteq C)$ (respectively $({\sim}C \sqsubseteq {\sim}D)$ and $({\sim}D \sqsubseteq {\sim}C)$). $\mathcal{I}$ is a model of $(C \equiv D)$ iff $\mathcal{I}$ is a model of $(C =^{+} D)$ and $(C =^{-} D)$. Finally, $\mathcal{I}$ is said to be a model of a TBox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) iff $\mathcal{I}$ is a model of every $\alpha \in \mathcal{T}$. The TBox $\mathcal{T}$ logically implies the TBox statement α (in symbols $\mathcal{T} \models \alpha$) iff every model of $\mathcal{T}$ is a model of $\{\alpha\}$.

Definition 5. A knowledge base Σ is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a model for Σ iff $\mathcal{I}$ is a model for both $\mathcal{T}$ and $\mathcal{A}$. Let α be an ABox statement or a TBox statement. The knowledge base Σ logically implies α (in symbols $\Sigma \models \alpha$) iff every model of Σ is a model of $\{\alpha\}$.

If α is an ABox or a TBox statement and $\mathcal{I}$ is a model of α, we shall write $\mathcal{I} \models \alpha$.

Proposition 6. *The system $\mathcal{CALC}^{\mathsf{C}}$ has the disjunction property and the constructible falsity property*:

(i) $\models_{\mathcal{CALC}^{\mathsf{C}}} (C \sqcup D) \Rightarrow \models_{\mathcal{CALC}^{\mathsf{C}}} C$ *or* $\models_{\mathcal{CALC}^{\mathsf{C}}} D$;
(ii) $\models_{\mathcal{CALC}^{\mathsf{C}}} {\sim}(C \sqcap D) \Rightarrow \models_{\mathcal{CALC}^{\mathsf{C}}} {\sim}C$ *or* $\models_{\mathcal{CALC}^{\mathsf{C}}} {\sim}D$.

A concept description is in *negation normal form* (*nnf*) if it contains $\sim$ only in front of concept names. Let $\forall r.^{d} = \exists r.$ and $\exists r.^{d} = \forall r.$. The following translation $\overline{(\cdot)}$ sends every concept description C to a concept description in *nnf*, where $A \in N_C$, $\diamond \in \{\sqcap, \sqcup, \supseteq\}$ and $\sharp \in \{\forall r., \exists r.\}$:

$$\begin{array}{rclrcl}\overline{A} & = & A & \overline{\sim A} & = & \sim A\\ \overline{\sim\sim C} & = & \overline{C} & \overline{C \diamond D} & = & \overline{C} \diamond \overline{D}\\ \overline{\sim (C \sqcap D)} & = & \overline{\sim C} \sqcup \overline{\sim D} & \overline{\sim (C \sqcup D)} & = & \overline{\sim C} \sqcap \overline{\sim D}\\ \overline{\sim (C \supseteq D)} & = & \overline{C} \sqcap \overline{\sim D} & & & \\ \overline{\sharp C} & = & \sharp \overline{C} & \overline{\sim \sharp C} & = & \sharp^{d} \overline{\sim C}.\end{array}$$

The notion of *nnf* extends in a natural way to ABox and TBox statements. If $C(a)$ is an ABox statement, its *nnf* $\overline{C(a)}$ is defined as $\overline{C}(a)$. The *nnf* of a TBox statement is defined as follows.

$$\begin{aligned}
\overline{C \sqsubseteq D} &=_{def} \overline{C} \sqsubseteq \overline{D},\\
\overline{C =^{+} D} &=_{def} \overline{C} =^{+} \overline{D},\\
\overline{C =^{-} D} &=_{def} \overline{\sim C} =^{+} \overline{\sim D},\\
\overline{C \equiv D} &=_{def} \{\overline{C =^{+} D},\ \overline{C =^{-} D}\}.
\end{aligned}$$

Proposition 7. *For every concept description C, and every interpretation $\mathcal{I}$, $\overline{C}$ is in nnf, and $(C =^{+} \overline{C})^{\mathcal{I}} = \Delta^{\mathcal{I}}$.*

Proof. It easy to see that for any concept description C and D, the relation $(C =^{+} D)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ is equivalent to $C^{\mathcal{I}} = D^{\mathcal{I}}$. The equality $C^{\mathcal{I}} = (\overline{C})^{\mathcal{I}}$ can be established via an induction on the complexity of concept descriptions. □

Note that the relation "$=^{+}$" cannot be replaced by "$=^{-}$". The equality $(C =^{-} \overline{C})^{\mathcal{I}} \Delta^{\mathcal{I}}$ is equivalent to $(\sim C)^{\mathcal{I}} = (\sim \overline{C})^{\mathcal{I}}$. Consider the concept $\sim (C \supseteq D)$, where $C, D \in N_C$. We have $\overline{\sim (C \supseteq D)} = C \sqcap \sim D$. According to Table 1 we have $(\sim\sim (C \supseteq D))^{\mathcal{I}} = (C \supseteq D)^{\mathcal{I}}$. At the same time $(\sim (C \sqcap \sim D))^{\mathcal{I}} = (\sim C)^{\mathcal{I}} \cup D^{\mathcal{I}}$. Let $\mathcal{I}$ be an interpretation such that $\Delta^{\mathcal{I}} = \{x\}$, $C^{\mathcal{I}} = (\sim C)^{\mathcal{I}} = \{x\}$, $D^{\mathcal{I}} = \varnothing$. Then $(C \supseteq D)^{\mathcal{I}} = \varnothing$ and $\sim C^{\mathcal{I}} \cup D^{\mathcal{I}} = \{x\}$.

4. The notion of a tableau

The tableau algorithms presented in [8] test for satisfiability. If classical negation $\neg$ is present, and subsumption is defined in terms of material implication $\neg C \sqcup D$, subsumption testing can be reduced to satisfiability testing: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable. In our paraconsistent framework, the satisfiability problem is trivial. It was shown in [12] that for any knowledge base Σ there is an interpretation $\mathcal{I}$ modelling Σ. Due to this reason, we construct a tableau algorithm testing for $\mathcal{CALC}^{\mathrm{C}}$-validity. Recall that we have defined the subsumption of C by D as the validity of $C \supseteq D$. As in [12], we use signed ABox and TBox statements, where the sign $+$ stands for "true" and the sign $-$ stands for "not true". Due to the lack of the law of excluded middle in our basic logic, non-truth cannot be expressed with the help of negation $\sim$. Thus, $-A(a)$ is not an abbreviation for $+ \sim A(a)$. A set of signed ABox statements is called a signed ABox. Since every TBox statement is equivalent to a finite number of TBox statements $C \sqsubseteq D$, we shall consider only signed TBox statements of this shape.

According to Proposition 7 the validity of a concept C is equivalent to the validity of its *nnf* $\overline{C}$. Due to this reason, our tableau algorithm works only with concepts in *nnf*. Note that reducing to *nnf* takes linear time.

Every tableau proof procedure is based on the search for a countermodel for a formula which should be proven. The proof has a tree structure and each branch of this tree corresponds to an attempt to construct a countermodel. If every attempt to construct a countermodel leads to a contradiction (in traditional tableau language, the corresponding branch of proof contains a *clash*, or is *closed*), the formula is proved. If at least one branch of the proof tree has no contradictions, we can reconstruct from this branch a countermodel of the original formula.

In this section we define the notion of a tableau. A tableau is a sort of partial interpretation, which is very close to the structure arising in the branch of a tableau proof procedure defined in the next section. Moreover, we prove that the tableau semantics is equivalent to the interpretation semantics.

We define the set of subconcepts $Sub(D)$ for $D \in Con(N_C, N_r)$ in *nnf*:

$$\begin{array}{ll}
Sub(A) := \{A\}, & A \in N_C;\\
Sub(\sim A) := \{\sim A\}, & A \in N_C;\\
Sub(C * D) := \{C * D\} \cup Sub(C) \cup Sub(D), & * \in \{\sqcup, \sqcap, \supseteq\};\\
Sub(Qr.C) := \{Qr.C\} \cup Sub(C), & Q \in \{\forall, \exists\}.
\end{array}$$

Then we extend this notion to TBox statements:

$$Sub(C \sqsubseteq D) := Sub(C \supseteq D).$$

Let Γ be a set of ABox and TBox statements in *nnf*.

$$Sub(\Gamma) := \bigcup_{D \in \Gamma} Sub(D).$$

For a set Γ of signed ABox and TBox statements in *nnf*, we put

$$\Gamma^{()} := \{D \mid +D \in \Gamma \text{ or } -D \in \Gamma\}$$

and

$$Sub(\Gamma) := \{+D, -D \mid D \in Sub(\Gamma^{()})\}.$$

Definition 8. Let Γ be a set of signed ABox and TBox statements in *nnf* and R_Γ be the set of role names occurring in Γ plus the new role name $\preccurlyeq$. A *tableau* T for Γ is a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$, where $\mathbf{S}$ is a set of individuals and $\mathcal{L} : \mathbf{S} \to 2^{Sub(\Gamma)}$ is an assignment that maps every individual to a set of concepts true or false of this individual. Finally, the map $\mathcal{E} : R_\Gamma \to 2^{\mathbf{S}\times\mathbf{S}}$ assigns to a role name its interpretation on $\mathbf{S}$. The set $\mathbf{S}$ and the mapping $\mathcal{L}$ are such that if an ABox statement $+(-)D(a)$ belongs to Γ, then $a \in \mathbf{S}$ and $+(-)D \in \mathcal{L}(a)$; if $+r(a,b) \in \Gamma$, then $a, b \in \mathbf{S}$ and $(a,b) \in \mathcal{E}(r)$; if $-r(a,b) \in \Gamma$, then $a, b \in \mathbf{S}$ and $(a,b) \notin \mathcal{E}(r)$; if a TBox statement $+(C \sqsubseteq D)$ is in Γ, then $+(C \supseteq D) \in \mathcal{L}(a)$ for all $a \in \mathbf{S}$; if $-(C \sqsubseteq D) \in \Gamma$, then there exists an individual $a \in \mathbf{S}$ such that $-(C \supseteq D) \in \mathcal{L}(a)$. Moreover, the following holds for any $a, b, c \in \mathbf{S}$, $C, D, E \in Sub(\Gamma^{()})$ and $r \in R_\Gamma$:

(1) $\{+C, -C\} \nsubseteq \mathcal{L}(a)$;

(2) a) if $+(C \sqcap D) \in \mathcal{L}(a)$, then $\{+C, +D\} \subseteq \mathcal{L}(a)$;
b) if $-(C \sqcap D) \in \mathcal{L}(a)$, then $-C \in \mathcal{L}(a)$ or $-D \in \mathcal{L}(a)$;

(3) a) if $+(C \sqcup D) \in \mathcal{L}(a)$, then $+C \in \mathcal{L}(a)$ or $+D \in \mathcal{L}(a)$;
b) if $-(C \sqcup D) \in \mathcal{L}(a)$, then $\{-C, -D\} \subseteq \mathcal{L}(a)$;

(4) a) if $+(C \supseteq D) \in \mathcal{L}(a)$ and $C \in N_C$ or $C = Qr.C'$, $Q \in \{\forall, \exists\}$, then $-C \in \mathcal{L}(a)$ or $+D \in \mathcal{L}(a)$;
b) if $+((C \sqcap D) \supseteq E) \in \mathcal{L}(a)$, then $+(C \supseteq (D \supseteq E)) \in \mathcal{L}(a)$;
c) if $+((C \sqcup D) \supseteq E) \in \mathcal{L}(a)$, then $+(C \supseteq E) \in \mathcal{L}(a)$ and $+(D \supseteq E) \in \mathcal{L}(a)$;
d) if $+((C \supseteq D) \supseteq E) \in \mathcal{L}(a)$, then $+E \in \mathcal{L}(a)$ or there exists $b \in \mathbf{S}$ such that $(a,b) \in (\mathcal{E}(\preccurlyeq))^+$ and $\{+C, -D, +(D \supseteq E)\} \subseteq \mathcal{L}(b)$, where $(\mathcal{E}(\preccurlyeq))^+$ is reflexive and transitive closure of $\mathcal{E}(\preccurlyeq)$;
e) if $-(C \supseteq D) \in \mathcal{L}(a)$, then there exists $b \in \mathbf{S}$ such that $(a,b) \in (\mathcal{E}(\preccurlyeq))^+$ and $\{+C, -D\} \subseteq \mathcal{L}(b)$;

(5) a) if $+\forall r.C \in \mathcal{L}(a)$, then $\forall b \in \mathbf{S}((a,b) \in (\mathcal{E}(\preccurlyeq))^+ \Rightarrow \forall c \in \mathbf{S}((b,c) \in \mathcal{E}(r) \Rightarrow +C \in \mathcal{L}(c)))$;
b) if $-\forall r.C \in \mathcal{L}(a)$, then $\exists b, c \in \mathbf{S}((a,b) \in (\mathcal{E}(\preccurlyeq))^+, (b,c) \in \mathcal{E}(r)$ and $-C \in \mathcal{L}(c))$;

(6) a) if $+\exists r.C \in \mathcal{L}(a)$, then $\exists b \in \mathbf{S}((a,b) \in \mathcal{E}(r)$ and $+C \in \mathcal{L}(b))$;
b) if $-\exists r.C \in \mathcal{L}(a)$, then $\forall b \in \mathbf{S}((a,b) \in \mathcal{E}(r) \Rightarrow -C \in \mathcal{L}(b))$;

(7) if $+C \in \mathcal{L}(a)$ and $(a,b) \in \mathcal{E}(\preccurlyeq)$, then $+C \in \mathcal{L}(b)$;

(8) if $(a,b) \in \mathcal{E}(\preccurlyeq)$ and $(a,c) \in \mathcal{E}(r)$, then there exists $d \in \mathbf{S}$ such that $(c,d) \in \mathcal{E}(\preccurlyeq)$ and $(b,d) \in \mathcal{E}(r)$.

Condition 1 of the above definition means that no concept C can be true and not true for an individual simultaneously, just as for an interpretation $\mathcal{I}$ and $a \in \Delta^\mathcal{I}$, if $a \in C^\mathcal{I}$, then it impossible that $a \notin C^\mathcal{I}$. The definition admits, however, that $\{+C, + \sim C\} \subseteq \mathcal{L}(a)$, just as the definition of an interpretation does not exclude the possibility that $C^\mathcal{I} \cap (\sim C)^\mathcal{I} \neq \varnothing$. Condition 7 corresponds to the property of interpretations established in Proposition 1, and Condition 8 to the property (P3) of interpretations. The other conditions, except of Condition 4, directly correspond to the definition of $C^\mathcal{I}$ in Table 1. The form of Condition 4 is determined by the rules for treating intuitionistic implication in the tableau algorithm (see Section 5). At first glance, it differs essentially from the definition of $(C \supseteq D)^\mathcal{I}$. However, we prove that the semantics of tableaux is equivalent to that of interpretations.

Lemma 9. *Let $\mathcal{I}$ be an arbitrary interpretation and $A, B, C \in Con(N_C, N_r)$. The following relations hold*:

(1) $((A \sqcap B) \supseteq C)^{\mathcal{I}} = (A \supseteq (B \supseteq C))^{\mathcal{I}}$;
(2) $((A \sqcup B) \supseteq C)^{\mathcal{I}} = ((A \supseteq C) \sqcap (B \supseteq C))^{\mathcal{I}}$;
(3) $((A \supseteq B) \supseteq C)^{\mathcal{I}} \subseteq (B \supseteq C)^{\mathcal{I}}$.

The *proof* of all items easily follows from the definition of interpretation of complex concept descriptions. □

Theorem 10. *For a knowledge base Σ and an ABox or TBox statement α, $\Sigma \not\models_{\mathcal{CALC}^{\mathrm{C}}} \alpha$ if and only if there exists a tableau for the set $\Gamma = +\Sigma \cup \{-\alpha\}$, where $+\Sigma = \{+\beta \mid \beta \in \Sigma\}$.*

Proof. Let $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for $\Gamma = +\Sigma \cup \{-\alpha\}$. We define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \preccurlyeq^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows

$$\begin{array}{ll} \Delta^{\mathcal{I}} := \mathbf{S} & \\ A^{\mathcal{I}} := \{a \mid +A \in \mathcal{L}(a)\}, & A \in N_C \cap Sub(\Gamma^0) \\ (\sim A)^{\mathcal{I}} := \{a \mid + \sim A \in \mathcal{L}(a)\}, & A \in N_C, \sim A \in Sub(\Gamma^0) \\ r^{\mathcal{I}} := \mathcal{E}(r), & r \in R_\Gamma \\ \preccurlyeq^{\mathcal{I}} := \text{reflexive and transitive closure of} & \mathcal{E}(\preccurlyeq). \end{array}$$

Item 7 in the tableau definition guarantees that $A^{\mathcal{I}}$ and $(\sim A)^{\mathcal{I}}$ satisfy conditions (P1) and (P2) respectively, whereas Item 8 implies that $(\preccurlyeq^{\mathcal{I}})^{-1} r^{\mathcal{I}} \subseteq r^{\mathcal{I}} (\preccurlyeq^{\mathcal{I}})^{-1}$ for all $r \in R_\Gamma$. Thus, $\mathcal{I}$ is really an interpretation.

Using the definition of tableaux and an induction on the structure of concept descriptions, we prove that for any $a \in \mathbf{S}$ and $C \in Sub(\Gamma^0)$ the following implications are true:

$$+C \in \mathcal{L}(a) \Rightarrow a \in C^{\mathcal{I}}$$
$$-C \in \mathcal{L}(a) \Rightarrow a \notin C^{\mathcal{I}}.$$

From these we immediately obtain that $\mathcal{I}$ models Σ but not α.

We check only the basis of the induction and the non-trivial case of implication.

If C is equal A or $\sim A$, where $A \in N_C$, then the first implication follows directly from the definition of interpretation $\mathcal{I}$. Let $-C \in \mathcal{L}(a)$, then $+C \notin \mathcal{L}(a)$ since the tableau T is clash-free. Again by definition of $\mathcal{I}$ we have $a \notin C^{\mathcal{I}}$.

For concepts of the form $C \supseteq D$ we prove the first implication by induction on the complexity of C.

a) Let $+(C \supseteq D) \in \mathcal{L}(a)$ and $C \in N_C$ or $C = Qr.C'$, $Q \in \{\forall, \exists\}$. By Item 7 in the tableau definition we have $+(C \supseteq D) \in \mathcal{L}(b)$ for any b such that $(a, b) \in (\mathcal{E}(\preccurlyeq))^+$. By Item 4 a) either $-C \in \mathcal{L}(b)$ or $+D \in \mathcal{L}(b)$. By induction hypotheses we have $b \notin C^{\mathcal{I}}$ or $b \in D^{\mathcal{I}}$ for any b with $a \preccurlyeq^{\mathcal{I}} b$, i.e. $a \in (C \supseteq D)^{\mathcal{I}}$.

b) If $+((C \sqcap D) \supseteq E) \in \mathcal{L}(a)$, then by Item 4 b) of the tableau definition we have $+(C \sqsubseteq (D \supseteq E)) \in \mathcal{L}(a)$. We apply the induction hypothesis to C, which is less complex than $C \sqcap D$, and obtain $a \in (C \sqsubseteq (D \supseteq E))^{\mathcal{I}}$. The latter is equivalent to $a \in ((C \sqcap D) \supseteq E)^{\mathcal{I}}$ by Lemma 9.

c) If $+((C \sqcup D) \supseteq E) \in \mathcal{L}(a)$, then $+(C \supseteq E), +(D \supseteq E) \in \mathcal{L}(a)$ by Item 4 c) of the tableau definition. Applying the induction hypothesis to C and D we obtain $a \in ((C \supseteq E) \sqcap (D \supseteq E))^{\mathcal{I}}$, which is equivalent to $a \in ((C \sqcup D) \supseteq E)^{\mathcal{I}}$ by Lemma 9.

d) Assume $+((C \supseteq D) \supseteq E) \in \mathcal{L}(a)$. If $+E \in \mathcal{L}(a)$, then $a \in E^{\mathcal{I}}$ by the induction hypothesis and $a \in ((C \supseteq D) \supseteq E)^{\mathcal{I}}$ by the definition of interpretations.

Assume that $+E \notin \mathcal{L}(a)$ but $a \notin ((C \supseteq D) \supseteq E)^{\mathcal{I}}$. Then there exists $b \in \mathbf{S}$ such that $a \preccurlyeq^{\mathcal{I}} b$ and $b \in (C \supseteq D)^{\mathcal{I}}$ and $b \notin E^{\mathcal{I}}$. The latter means that $+E \notin \mathcal{L}(b)$.

If $+E \notin \mathcal{L}(b)$, then there exists $c \in \mathbf{S}$ such that $b \preccurlyeq^{\mathcal{I}} c$ and $\{+C, -D\} \subseteq \mathcal{L}(c)$. By the induction hypotheses we have $c \in C^{\mathcal{I}}$ and $c \notin D^{\mathcal{I}}$. Consequently, $b \notin (C \supseteq D)^{\mathcal{I}}$ and we arrive at a contradiction.

e) If $-(C \supseteq D) \in \mathcal{L}(a)$, then there exists $b \in \mathbf{S}$ such that $a \preccurlyeq^{\mathcal{I}} b$ and $\{+C, -D\} \in \mathcal{L}(b)$. By the induction hypotheses $b \in C^{\mathcal{I}}$ and $b \notin D^{\mathcal{I}}$, i.e. $a \notin (C \supseteq D)^{\mathcal{I}}$.

Conversely. Assume $\Sigma \not\models_{\mathcal{CALC}^{\mathsf{C}}} \alpha$. Let $\Gamma = +\Sigma \cup \{-\alpha\}$ and $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \Sigma$ and $\mathcal{I} \not\models \alpha$. The tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is defined as follows.

$$\mathbf{S} := \Delta^{\mathcal{I}}, \quad \mathcal{E}(\preccurlyeq) := \preccurlyeq^{\mathcal{I}}, \quad \mathcal{E}(r) := r^{\mathcal{I}}, \quad r \in R_{\Gamma},$$
$$\mathcal{L}(a) := \{+C \mid a \in C^{\mathcal{I}}, C \in Sub(\Gamma^0)\} \cup \{-C \mid a \notin C^{\mathcal{I}}, C \in Sub(\Gamma^0)\}, \quad a \in \Delta^{\mathcal{I}}.$$

It remains to be proved that T is a tableau for Γ. In fact, the only non-trivial item to check is Item 4 in the tableau definition.

a) If $+(C \supseteq D) \in \mathcal{L}(a)$ and $C \in N_C$ or $C = Qr.C'$, $Q \in \{\forall, \exists\}$, then $a \in (C \supseteq D)^{\mathcal{I}}$ by definition of T. According to the definition of interpretation we have that $a \notin C^{\mathcal{I}}$ or $a \in D^{\mathcal{I}}$, i.e. $-C \in \mathcal{L}(a)$ or $+D \in \mathcal{L}(a)$.

b) If $+((C \sqcap D) \supseteq E) \in \mathcal{L}(a)$, then $a \in ((C \sqcap D) \supseteq E)^{\mathcal{I}} = (C \supseteq (D \supseteq E))^{\mathcal{I}}$. The last equality is due to Item 1 of Lemma 9. The definition of T implies $+(C \supseteq (D \supseteq E)) \in \mathcal{L}(a)$.

c) If $+((C \sqcup D) \supseteq E) \in \mathcal{L}(a)$, then $a \in ((C \sqcup D) \supseteq E)^{\mathcal{I}}$. By Item 2 of Lemma 9 this is equivalent to $a \in (C \supseteq E)^{\mathcal{I}}$ and $a \in (D \supseteq E)^{\mathcal{I}}$, i.e. $+(C \supseteq E) \in \mathcal{L}(a)$ and $+(D \supseteq E) \in \mathcal{L}(a)$.

d) If $+((C \supseteq D) \supseteq E) \in \mathcal{L}(a)$, then $a \in ((C \supseteq D) \supseteq E)^{\mathcal{I}}$. By Item 3 of Lemma 9 we have $a \in (D \supseteq E)^{\mathcal{I}}$. If $+E \notin \mathcal{L}(a)$, i.e. $a \notin E^{\mathcal{I}}$, then $a \notin (C \supseteq D)^{\mathcal{I}}$, and there exists $b \in \Delta^{\mathcal{I}}$ such that $a \preccurlyeq^{\mathcal{I}} b$, $b \in C^{\mathcal{I}}$, and $b \notin D^{\mathcal{I}}$. Since $a \preccurlyeq^{\mathcal{I}} b$ we have by Proposition 1 that $b \in (D \supseteq E)^{\mathcal{I}}$. In this way, we proved $\{+C, -D, +(D \supseteq E)\} \subseteq \mathcal{L}(b)$ and $(a, b) \in \mathcal{E}(\preccurlyeq)$.

e) If $-(C \supseteq D) \in \mathcal{L}(a)$, then $a \notin (C \supseteq D)^{\mathcal{I}}$ and by definition of $(C \supseteq D)^{\mathcal{I}}$ there exists $b \in \Delta^{\mathcal{I}}$ such that $a \preccurlyeq^{\mathcal{I}} b$, $b \in C^{\mathcal{I}}$, and $b \notin D^{\mathcal{I}}$. In other words, $\{+C, -D\} \subseteq \mathcal{L}(b)$. □

5. Tableau algorithm

Now we describe the tableau algorithm for $\mathcal{CALC}^{\mathsf{C}}$ which falsifies concepts in *nnf*. For any concept D this algorithm works on a *completion tree*. This is a tree where each node x is labelled with a set $\mathcal{L}(x) \subseteq Sub(\Gamma)$, where $\Gamma = \{-D\}$, and each edge (x, y) is labelled with an element $\mathcal{L}((x, y))$ of R_{Γ}. The algorithm expands the tree either by extending $\mathcal{L}(x)$ for some node x or by adding new leaf nodes. Some elements of sets $\mathcal{L}(x)$ we mark additionally with the sign $\downarrow$ and call them *passive*. All other formulas are *active*. We apply expansion rules only to active formulas and only if the application adds some new elements to the set of active formulas. We assume also that $\{+(-)A, +(-)A\downarrow\} = \{+(-)A\downarrow\}$.

Let x and y be nodes in $\mathbf{T}$. We say that x is a *predecessor* of y and y is a *successor* of x if x and y are connected by an edge (x, y). If this edge is labelled with an element γ of R_{Γ}, then x is a γ*-predecessor* of y and y is a γ*-successor* of x. We say that a node x is an R*-predecessor* (R*-successor*) of y if x is an r-predecessor (r-successor) of y for some role name r. The *ancestor* relation is the transitive closure of predecessor and the *descendant* relation is the transitive closure of successor. The $\preccurlyeq$-(R-)*ancestor* relation is the transitive closure of $\preccurlyeq$-(R-)predecessor and the $\preccurlyeq$-(R-)*descendant* relation is the transitive closure of $\preccurlyeq$-(R-)successor.

For a node x of $\mathbf{T}$, we denote by $\mathbf{T}(x)$ the *subtree* of $\mathbf{T}$ generated by x consisting of x, all its descendants and all edges connecting these elements. The $\preccurlyeq$-(R-)*subtree* of $\mathbf{T}$ generated by a, $\mathbf{T}_{\preccurlyeq}(a)$ ($\mathbf{T}_R(a)$), consists of a, all its $\preccurlyeq$-(R-) descendants and all edges connecting these elements marked with $\preccurlyeq$ (with r for some role $r \in R_{\Gamma}$).

The algorithm initializes the construction of a completion tree $\mathbf{T}$ with an initial tree $\mathbf{T}_0$ consisting of a unique node labelled with $\{-D\}$. Further, the tree is expanded by repeatedly applying the rules from Table 2. In this table, $a \preccurlyeq b$ means that a is a $\preccurlyeq$-ancestor of b or $a = b$, C, D, and E stand for arbitrary concepts. We write arb as an abbreviation of $\mathcal{L}((a, b)) = r$ and $(\mathcal{L}(a))^+ := \{+C \mid +C \in \mathcal{L}(a)\}$. The new nodes created by a rule are marked with *. For example, if we apply rule $(\supseteq -)$ to a concept $-(C \supseteq D)$ from the labelling of a node a, then we add to the completion tree a new leaf node b, a $\preccurlyeq$-successor of a, and label b with $(\mathcal{L}(a))^+ \cup \{+C, -D\}$; applying rule $(\forall -)$ to $-\forall r.C$ from $\mathcal{L}(a)$ we add to the completion tree two new nodes: a $\preccurlyeq$-successor b of a marked with $(\mathcal{L}(a))^+$ and an r-successor c of b labelled with $\{-C\}$.

Note that each of the following pairs of rules: $(\supseteq\supseteq a)$ and $(\supseteq\supseteq s)$, $(\supseteq \forall a)$ and $(\supseteq \forall s)$, $(\supseteq \exists a)$ and $(\supseteq \exists s)$ can be considered as one non-deterministic rule.

The rules have different priorities. The rules $(\sqcap+)$, $(\sqcap-)$, $(\sqcup+)$, $(\sqcup-)$, $(\supseteq N)$, $(\supseteq \sqcup)$, $(\supseteq \sqcap)$, $(\supseteq\supseteq s)$, $(\supseteq \forall a)$, $(\supseteq \forall s)$, $(\supseteq \exists a)$, $(\supseteq \exists s)$ are of the highest priority, 1. All rules of this group expand labellings of existing nodes and create no new nodes. The next priority, 2, has the rule $(\preccurlyeq^{-1} r)$. This rule lifts R-successors of nodes up to their $\preccurlyeq$-successors. The rules $(\forall+)$, $(\exists+)$, $(\exists-)$ have priority 3. These rules expand R-subtrees, not $\preccurlyeq$-subtrees and transfer

Table 2
Expansion rules for $\mathcal{CALC}^{\mathsf{C}}$

$(\sqcap+)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(C\sqcap D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+C,+D,+(C\sqcap D)\downarrow\}}$	$(\sqcap-)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-(C\sqcap D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{-E,-(C\sqcap D)\downarrow\},\ E\in\{C,D\}}$
$(\sqcup+)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(C\sqcup D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+E,+(C\sqcup D)\downarrow\},\ E\in\{C,D\}}$	$(\sqcup-)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-(C\sqcup D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{-C,-D,-(C\sqcup D)\downarrow\}}$
$(\supseteq N)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+C,+(C\supseteq D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+C,+D,+(C\supseteq D)\downarrow\}}$		
$(\supseteq -)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-(C\supseteq D)\}}{a\preccurlyeq b^*,\ \mathcal{L}(a):=\mathcal{L}'\cup\{-(C\supseteq D)\downarrow\},\ \mathcal{L}(b^*):=(\mathcal{L}(a))^+\cup\{+C,-D\}}$		
$(\supseteq \sqcup)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+((C\sqcup D)\supseteq E)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+(C\supseteq E),+(D\supseteq E),+((C\sqcup D)\supseteq E)\downarrow\}}$		
$(\supseteq \sqcap)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+((C\sqcap D)\supseteq E)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+(C\supseteq(D\supseteq E)),+((C\sqcap D)\supseteq E)\downarrow\}}$		
$(\supseteq\supseteq a)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+((C\supseteq D)\supseteq E)\},\ \neg\exists b(a\preccurlyeq b,\ \{+(D\supseteq E),+C,-D\}\subseteq\mathcal{L}(b))}{a\preccurlyeq b^*,\ \mathcal{L}(b^*):=(\mathcal{L}')^+\cup\{+(D\supseteq E),+C,-D,+((C\supseteq D)\supseteq E)\downarrow\}}$		
$(\supseteq\supseteq s)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+((C\supseteq D)\supseteq E)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+E,+((C\supseteq D)\supseteq E)\downarrow\}}$		
$(\supseteq \forall a)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(\forall r.C\supseteq D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{-\forall r.C,+(\forall r.C\supseteq D)\}}$	$(\supseteq \forall s)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(\forall r.C\supseteq D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+D,+(\forall r.C\supseteq D)\downarrow\}}$
$(\supseteq \exists a)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(\exists r.C\supseteq D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{-\exists r.C,+(\exists r.C\supseteq D)\}}$	$(\supseteq \exists s)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+(\exists r.C\supseteq D)\}}{\mathcal{L}(a):=\mathcal{L}'\cup\{+D,+(\exists r.C\supseteq D)\downarrow\}}$
$(\forall+)$	$\dfrac{+\forall r.C\in\mathcal{L}(a),\ arb,\ +C\notin\mathcal{L}(b)}{\mathcal{L}(b):=\mathcal{L}(b)\cup\{+C\}}$		
$(\forall-)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-\forall r.C\},\ \neg\exists b,c(a\preccurlyeq b,\ brc,\ -C\in\mathcal{L}(c))}{a\preccurlyeq b^*,\ b^*rc^*,\ \mathcal{L}(a):=\mathcal{L}'\cup\{-\forall r.C\downarrow\},\ \mathcal{L}(b^*):=(\mathcal{L}(a))^+,\ \mathcal{L}(c^*):=\{-C\}}$		
$(\exists+)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{+\exists r.C\},\ \neg\exists b(arb,\ +C\in\mathcal{L}(b))}{arb^*,\ \mathcal{L}(a):=\mathcal{L}'\cup\{+\exists r.C\downarrow\},\ \mathcal{L}(b^*):=\{+C\}}$		
$(\exists-)$	$\dfrac{-\exists r.C\in\mathcal{L}(a),\ arb,\ -C\notin\mathcal{L}(b)}{\mathcal{L}(b):=\mathcal{L}(b)\cup\{-C\}}$		
$(\preccurlyeq^{-1} r)$	$\dfrac{c\preccurlyeq a,\ crb,\ \neg\exists d(ard,\ b\preccurlyeq d)}{ard^*,\ b\preccurlyeq d^*,\ \mathcal{L}(d^*):=(\mathcal{L}(b))^+}$		

concepts from R-predecessors to R-successors. And the lowest priority, 4, have the rules $(\forall-)$, $(\supseteq\supseteq a)$ and $(\supseteq -)$. These rules expand $\preccurlyeq$-subtrees. We apply expansion rules according to their priority. The rules of lower priority can be applied, only when it is impossible to apply the rules of higher priority. The only exception is the rule $(\supseteq\supseteq s)$, which constitute together with $(\supseteq\supseteq a)$ one non-deterministic rule. Therefore, we can either apply this rule together with other rules of priority 1 or do not apply it and start to apply rules of lower priority.

The necessity to introduce priorities of rules is explained by conditions (P1)–(P3) in the definition of an interpretation. Every positively signed concept should be transferred from a node to each of its $\preccurlyeq$-descendants. To this end, first, we complete the expansion of the labelling $\mathcal{L}(a)$ with the help of rules of priorities 1 and 3, second, we create its $\preccurlyeq$-successors and transfer to them all positively signed concepts from $\mathcal{L}(a)$ via rules of priority 4. The rule $(\preccurlyeq^{-1} r)$ guaranties that (P3) is satisfies for the completion tree. The priority 2 of this rule prevents creating additional R-successors by the rules of priority 3.

A node x is called a $\supseteq$-predecessor of y, if x is a $\preccurlyeq$-predecessor of y and y arises as a result of an application of the rule $(\supseteq -)$ or $(\supseteq\supseteq a)$. The notions of $\supseteq$-successor, $\supseteq$-ancestor, $\supseteq$-descendant, and $\supseteq$-subtree are defined in an obvious manner. The $\supseteq$-*subtree* of $\mathbf{T}$ generated by a, $\mathbf{T}_{\supseteq}(a)$, consists of a, all its $\supseteq$-descendants and all edges connecting these elements marked with $\preccurlyeq$.

A completion tree $\mathbf{T}$ is said to contain a *clash* if, for a node x in $\mathbf{T}$ and a concept C, $\{+C,-C\}\subseteq\mathcal{L}(x)$. If an application of one of the expansion rules yields a clash, the construction of the completion tree terminates at that moment.

A completion tree is *complete* when for some node a, $\mathcal{L}(a)$ contains a clash or when none of the rules are applicable. If, for an input set $\Gamma=\{-D\}$, the expansion rules can be applied so that they yield a complete, clash-free completion tree, then we claim that D can be refuted at some interpretation. Otherwise, D is $\mathcal{CALC}^{\mathsf{C}}$-valid. Note that a completion tree may be infinite.[1]

Proposition 11 *(Soundness). If the expansion rules can be applied to* $\{-D\}$ *so that they yield a complete and clash-free completion tree, then there exists a tableau for* $\{-D(a)\}$*, where* a *is the root of the completion tree.*

Proof. Let $\mathbf{T}$ be the complete and clash-free completion tree obtained by applying the tableau algorithm to $\{-D\}$, and let $\mathcal{L}$ be the labelling of nodes in $\mathbf{T}$. The tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ is defined as follows:

$$\mathbf{S} = \{x \mid x \text{ is a node in } \mathbf{T}\}$$
$$x\mathcal{E}(\gamma)y \quad \text{iff} \quad (x, y) \text{ is an edge labelled by } \gamma \text{ in } \mathbf{T}, \gamma \in R_{\{-D\}}.$$

Moreover, we set $\mathcal{L}'(x) := \mathcal{L}(x)$ and then close $\mathcal{L}'(x)$ under the following rule:

[a] if $+C \supseteq D \in \mathcal{L}'(a)$ and $+D \notin \mathcal{L}'(a)$, then $-C \in \mathcal{L}'(a)$.

We check that T is indeed a tableau. Condition 1 of the tableau definition holds, because the tree $\mathbf{T}$ is clash free and the expansion rule $(\supseteq N)$ guaranties that the closure rule [a] does not introduce any clashes.

Condition 7 is satisfied, because every expansion rule introducing a $\preccurlyeq$-successor is such that every positively signed concept description is persistent either as an active or as a passive formula. This follows from the form of rules $(\supseteq -)$, $(\supseteq\supseteq a)$, $(\forall -)$ and $(\preccurlyeq^{-1} r)$. Moreover, rules $(\supseteq -)$, $(\supseteq\supseteq a)$ and $(\forall -)$ have the lowest priority. The latter means that if a $\preccurlyeq$-predecessor b of a node a was created at some stage, no new positively signed concept will be added to $\mathcal{L}(a)$ after this stage. Note that the closure rule [a] for $\mathcal{L}'$ introduces only negatively signed concept descriptions. The rule $(\preccurlyeq^{-1} r)$ has priority 2. If this rule is applied to nodes a, b and c, where $c \preccurlyeq a$ and $b \preccurlyeq d$, then at some previous step a was created by some rule of priority 4. Obviously, at this step the set $\mathcal{L}(b)$ is completely expanded.

Condition 8 is satisfied due to the rule $(\preccurlyeq^{-1} r)$ and the fact that $\mathbf{T}$ is complete.

Let us check that Conditions 2b), 3b), 4e), 5b), 6b) concerning negatively signed concept descriptions are satisfied. Note that satisfiability of these conditions for elements of $\mathcal{L}(a)$, $a \in \mathbf{S}$ follows directly from the fact that $\mathbf{T}$ is complete and from the form of the respective expansion rules. Let us check these conditions for concept descriptions added by closure rule [a].

2b) Let $+((C \sqcap D) \supseteq E) \in \mathcal{L}(a)$, $+E \notin \mathcal{L}(a)$ and $-(C \sqcap D) \in \mathcal{L}'(a)$. The rule $(\supseteq \sqcap)$ guaranties that $+(C \supseteq (D \supseteq E)) \in \mathcal{L}(a)$. If $+(D \supseteq E) \in \mathcal{L}(a)$, then in view of $+E \notin \mathcal{L}(a)$ we have $-D \in \mathcal{L}'(a)$, i.e. Condition 2b) is satisfied for $-(C \sqcap D) \in \mathcal{L}'(a)$. Assume $+(D \supseteq E) \notin \mathcal{L}(a)$, then $-C \in \mathcal{L}'(a)$ and Condition 2b) again is satisfied.

3b) Let $+((C \sqcup D) \supseteq E) \in \mathcal{L}(a)$, $+E \notin \mathcal{L}(a)$ and $-(C \sqcup D) \in \mathcal{L}'(a)$. By the rule $(\supseteq \sqcup)$ we have $+(C \supseteq E), +(D \supseteq E) \in \mathcal{L}(a)$. Since $+E \notin \mathcal{L}(a)$, we have $-C, -D \in \mathcal{L}'(a)$. This means that Condition 3b) is satisfied for $-(C \sqcup D) \in \mathcal{L}'(a)$.

4e) Let $+((C \supseteq D) \supseteq E) \in \mathcal{L}(a)$, $+E \notin \mathcal{L}(a)$ and $-(C \supseteq D) \in \mathcal{L}'(a)$. By the rule $(\supseteq\supseteq a)$ there is a $\preccurlyeq$-successor b of a such that $\{+C, -D, +(D \supseteq E)\} \subseteq \mathcal{L}(a)$. This means, in particular, that Condition 4e) is satisfied for $-(C \supseteq D) \in \mathcal{L}'(a)$.

5b) and 6b) If $+(Qr.C \supseteq D) \in \mathcal{L}(a)$, $Q \in \{\forall, \exists\}$, and $+D \notin \mathcal{L}(a)$, the rule $(\supseteq Qa)$ guarantees that $-Qr.C \in \mathcal{L}(a)$. Thus, the closure rule [a] does not add a new concept description in this case.

Now we have to check that Conditions 2a), 3a), 4 a)–d), 5a) and 6a) are satisfied.

2a) If $+(C \sqcap D) \in \mathcal{L}(a)$, then this concept description is inactive and it became inactive as a result of applying the rule $(\sqcap +)$ to $+(C \sqcap D) \in \mathcal{L}(b)$, where $a = b$ or b is a $\preccurlyeq$-ancestor of a. Therefore, $\{+C, +D\} \subseteq \mathcal{L}(b)$ and by Condition 7 $\{+C, +D\} \subseteq \mathcal{L}(a)$.

In a similar way one can check Conditions 3a), 4b), and 4c).

4a) If $+(C \supseteq D) \in \mathcal{L}(a)$ and $C = Qr.C'$, then $-C \in \mathcal{L}(a)$ or $+D \in \mathcal{L}(a)$ according to the rules $(\supseteq Qa)$ and $(\supseteq Qs)$, $Q \in \{\forall, \exists\}$. Assume $C \in N_C$. If $+C \in \mathcal{L}(a)$, then $+D \in \mathcal{L}(a)$ by the rule $(\supseteq N)$. If $+D \notin \mathcal{L}(a)$, then $-C \in \mathcal{L}'(a)$ by the closure rule [a].

[1] For an infinite completion tree $\mathbf{T}$, the condition to be complete means that for some node a, $\mathcal{L}(a)$ contains a clash or for any part $\mathbf{T}_i$ of $\mathbf{T}$ constructed up to stage i, there is a stage j such that none of the rules can be applied to $\mathbf{T}_i$ after this stage.

4d) If $+((C \supseteq D) \supseteq E) \in \mathcal{L}(a)$, then this concept description became inactive after an application of rules $(\supseteq\supseteq a)$ or $(\supseteq\supseteq s)$ to $+((C \supseteq D) \supseteq E) \in \mathcal{L}(b)$, where $a = b$ or b is a $\preccurlyeq$-ancestor of a. If $a = b$, then either $+E \in \mathcal{L}(a)$ according to rule $(\supseteq\supseteq s)$ or rule $(\supseteq\supseteq s)$ creates a $\preccurlyeq$-successor c such that $\{+C, -D, +(D \supseteq E)\} \subseteq \mathcal{L}(c)$. If $a \neq b$ and $+E \in \mathcal{L}(b)$, then $+E \in \mathcal{L}(a)$ by Condition 7. Otherwise, there is a $\preccurlyeq$-successor b' of b such that $\{+C, -D, +(D \supseteq E)\} \subseteq \mathcal{L}(b')$. Note that $+((C \supseteq D) \supseteq E)$ remains active in b and so it is active in all $\preccurlyeq$-successors of b different from b'. Therefore, a is a $\preccurlyeq$-descendant of b' and $\{+C, +(D \supseteq E)\} \subseteq \mathcal{L}(a)$. If $+E \notin \mathcal{L}(a)$, then $-D \in \mathcal{L}'(a)$ and Condition 4d) is satisfied for $+((C \supseteq D) \supseteq E) \in \mathcal{L}'(a)$.

5a) If $+\forall r.C \in \mathcal{L}(a)$, then the rule $(\forall-)$ guarantees that $+\forall r.C$ remains active all the time and for any r-successor b of a, $+C \in \mathcal{L}(a)$. If b is a $\preccurlyeq$-descendant of a, then $+\forall r.C \in \mathcal{L}(b)$ and for every r-successor c of b, $+C \in \mathcal{L}(c)$ holds due to one of the rules $(\forall+)$ or $(\preccurlyeq^{-1} r)$.

6a) This follows directly from the form of the rule $(\exists+)$. □

Proposition 12. *For any completion tree* $\mathbf{T}$ *and node* a *in* $\mathbf{T}$, *the construction of the subtrees* $\mathbf{T}_{\supseteq}(a)$ *and* $\mathbf{T}_R(a)$ *terminates.*

Proof. First we consider the subtree $\mathbf{T}_{\supseteq}(a)$. Define the $\supseteq$-weight $W_{\supseteq}(C)$ of a concept (description) C as follows.

$$W_{\supseteq}(A) = W_{\supseteq}(Qr.C) = 0, \quad \text{where } A \in N_C \text{ and } Q \in \{\forall, \exists\};$$
$$W_{\supseteq}(C \sqcup D) = W_{\supseteq}(C \supseteq D) = W_{\supseteq}(C) + W_{\supseteq}(D) + 1;$$
$$W_{\supseteq}(C \sqcap D) = W_{\supseteq}(C) + W_{\supseteq}(D) + 2.$$

For signed concepts we put

$$W_{\supseteq}(\epsilon C) = W_{\supseteq}(C), \quad \text{where } \epsilon \in \{+, -\}.$$

Obviously, signed concepts are preordered wrt their weights and this preordering is well-founded. Thus, we can define the well-founded Dershowitz–Manna preordering (see [5]) on sets of signed concepts as follows: $\Gamma \leqslant_{DM} \Delta$ iff Δ is obtained from Γ by replacing each element of Γ by zero or a finite number of signed concepts having strictly smaller weights.

Only rules of priority 1 and rules $(\supseteq\supseteq a)$ and $(\supseteq -)$ are involved in the construction of the $\supseteq$-subtree. Application of each of these rules except for $(\supseteq \forall a)$ and $(\supseteq \exists a)$ results in a set of active signed formulas strictly smaller wrt the Dershowitz–Manna preordering. Note that rules $(\supseteq \forall a)$ and $(\supseteq \exists a)$ are the only rules admitting duplication of active formulas, and application of these rules may result in a set of active formulas which is not strictly smaller wrt the Dershowitz–Manna preordering than the original set of active formulas.

However, let us consider a particular node b of $\mathbf{T}_{\supseteq}(a)$. In the construction of $\mathcal{L}(b)$ to each concept of the form $+Qr.C \supseteq D$ the rule $(\supseteq Qa)$, $Q \in \{\forall, \exists\}$ can be applied only once. And neither of the rules expanding $\mathcal{L}(b)$ applies to $-Qr.C$ resulting from the application of $(\supseteq Qa)$.

Since application of other rules yields a set of formulas smaller wrt $\leqslant_{DM}$, such application terminates. In this way, in the expansion of $\mathcal{L}(b)$ rules $(\supseteq Qa)$ can be applied only finitely many times. Thus, the expansion of $\mathcal{L}(b)$ terminates.

Denote by $\mathcal{L}^*(b)$ the set $\mathcal{L}(b)$ without concepts $-Qr.C$ resulting from the application of rules $(\supseteq Qa)$. It can be easily seen that rules $(\supseteq -)$ and $(\supseteq\supseteq a)$ can be applied only to elements of $\mathcal{L}^*(b)$ and such application creates a node c, $\supseteq$-successor of b with $\mathcal{L}(c)$ strictly smaller than $\mathcal{L}(b)$ wrt $\leqslant_{DM}$. Since $\leqslant_{DM}$ is well founded, each path in $\mathbf{T}_{\supseteq}(a)$ is finite. It remains to note that each element of $\mathbf{T}_{\supseteq}(a)$ has finitely many $\supseteq$-successors.

To prove that the construction of subtree $\mathbf{T}_R(a)$ terminates we define the concept of R-weight as follows.

$$W_R(A) = W_R(C \supseteq D) = 0, \quad \text{where } A \in N_C;$$
$$W_R(C \sqcup D) = W_R(C \sqcap D) = W_R(C) + W_R(D) + 1;$$
$$W_R(Qr.C) = W_R(C) + 1, \quad \text{where } Q \in \{\forall, \exists\}.$$

The proof is finished as above. □

Proposition 13 *(Completeness). If there exists a tableau for* $\{-D(a)\}$, *then the expansion rules can be applied in such a way that the tableau algorithm outputs a complete and clash-free completion tree for* $\{-D\}$.

Proof. Let $T = (\mathbf{S}, \mathcal{L}^*, \mathcal{E})$ be the given tableau for $\{-D(a)\}$. The root node to which the tableau algorithm is applied to obtain a complete and clash-free completion tree $\mathbf{T}$ is a, with $\mathcal{L}(a) = \{-D\}$. We define a function $\pi : \{x \mid x \text{ is a node in } \mathbf{T}\} \longrightarrow \mathbf{S}$ such that $\pi(a) = a$, $\mathcal{L}(x) \subseteq \mathcal{L}^*(\pi(x))$, $\mathcal{L}((x, y)) = r \in R_{\{-D\}}$ implies $(\pi(x), \pi(y)) \in \mathcal{E}(r)$, and $\mathcal{L}((x, y)) = \preccurlyeq$ implies $(\pi(x), \pi(y)) \in (\mathcal{E}(\preccurlyeq))^+$. In building up $\mathbf{T}$, we consider only the nondeterministic rules $(\sqcap -)$, $(\sqcup +)$, $(\supseteq\supseteq a)$, $(\supseteq\supseteq s)$, $(\supseteq \forall a)$, $(\supseteq \forall s)$, $(\supseteq \exists a)$, and $(\supseteq \exists s)$ and some rules creating new nodes.

Assume that a signed concept ϵB is active at a current stage and the respective rule has an appropriate priority to be applied at this stage. Tree $\mathbf{T}$ is expanded as follows:

If $\epsilon B = -(C \sqcap D) \in \mathcal{L}(x)$ and $\pi(x) = b$, then by definition of tableaux $-E \in \mathcal{L}^*(b)$ for $E \in \{C, D\}$. Apply to $-(C \sqcap D)$ the rule $(\sqcap -)$ adding $-E$ to $\mathcal{L}(x)$.

The case of $\epsilon B = +(C \sqcup D)$ and rule $(\sqcup +)$ is treated similarly.

If $\epsilon B = +\forall r.C \supseteq D \in \mathcal{L}(x)$ and $\pi(x) = b$, then $\mathcal{L}^*(b) \cap \{-\forall r.C, +D\} \neq \varnothing$. If $-\forall r.C \in \mathcal{L}^*(b)$, we apply the rule $(\supseteq \forall a)$, otherwise the rule $(\supseteq \forall s)$.

The case of $\epsilon B = +\exists r.C \supseteq D$ is treated similarly.

If $\epsilon B = -(C \supseteq D) \in \mathcal{L}(x)$ and $\pi(x) = b$, then by definition of tableaux there is $c \in \mathbf{S}$ such that $(b, c) \in (\mathcal{E}(\preccurlyeq))^+$ and $\{+C, -D\} \subseteq \mathcal{L}^*(c)$. We apply rule $(\supseteq -)$ creating a new node y and putting $\mathcal{L}((x, y)) = \preccurlyeq$ and $\pi(y) = c$.

If $\epsilon B = +((C \supseteq D) \supseteq E) \in \mathcal{L}(x)$ and $\pi(x) = b$, then either $+E \in \mathcal{L}(b)$, or there is $c \in \mathbf{S}$ with $(b, c) \in (\mathcal{E}(\preccurlyeq))^+$ and $\{+(D \supseteq E), +C, -D\} \subseteq \mathcal{L}^*(c)$ and $(b, c) \in \mathcal{E}(\preccurlyeq)$. In the first case, we apply rule $(\supseteq\supseteq s)$, in the second one, we apply rule $(\supseteq\supseteq a)$ and create a new node y with $\mathcal{L}((x, y)) = \preccurlyeq$ and $\pi(y) = c$.

If $\epsilon B = +\exists r.C \in \mathcal{L}(x)$ and $\pi(x) = b$, then there is an r-successor c of b such that $+C \in \mathcal{L}^*(c)$. Apply rule $(\exists +)$ creating a new node y such that $\mathcal{L}((x, y)) = r$ and $\pi(y) = c$.

Condition (1) in the tableau definition and the inclusion $\mathcal{L}(x) \subseteq \mathcal{L}^*(\pi(x))$ imply that $\mathbf{T}$ is clash-free. It remains to note that we can obtain a complete completion tree. It follows from Proposition 12 that at any stage the application of rules of priority 1 terminates after a finite number of stages. The number of subsequent applications of the rule $(\preccurlyeq^{-1})$ is obviously finite. Rules of priority 3 as well as rules of priority 1 extend subtrees of the form $\mathbf{T}_R(b)$. Thus, taking into account Proposition 12 we conclude again that at any stage application of rules of priorities 1–3 terminates after a finite number of stages. Finally, to guaranty that every rule of priority 4 will be applied at some stage it would be enough to agree that first we apply rules of priority 4 to concepts that arose at earlier stages. □

Unfortunately, we cannot prove that the tableau algorithm terminates. The duplication of formulas by the rules $(\supseteq Qa)$, $Q \in \{\forall, \exists\}$ combined with the rule $(\forall -)$ produces loops. Consider the following example. Let $\mathcal{L}(a_0) = \{+\forall r_0.A \supseteq B, +\forall r_1.C \supseteq D, -\forall r_0.A, -\forall r_1.C\}$, where A, B, C and D are concept names. Applying rule $(\forall -)$ alternatively to roles r_0 and r_1 and expanding labellings of nodes with the help of rule $(\supseteq \forall a)$, we obtain the following infinite completion tree, where $\mathcal{L}(a_i) = \mathcal{L}(a_0)$, $i = 1, 2, \ldots$, see Fig. 2.

An important feature of this loop is that all nodes of the ascending $\preccurlyeq$-chain have the same labelling, $\mathcal{L}(a_i) = \mathcal{L}(a_j)$ for all i and j. In fact, the tableau algorithm described above can produce only such loops. The above infinite tableau can be replaced by the finite one depicted in Fig. 3.

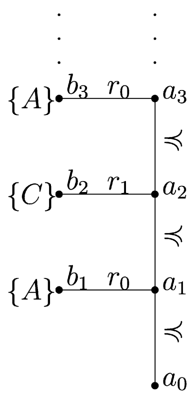

Fig. 2.

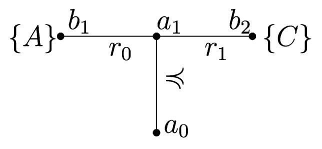

Fig. 3.

We can avoid such loops by replacing the rule $(\forall-)$ with three different rules presented in Table 3. The rule $(\forall-)$ can be applied only to concepts $-\forall r.C \in \mathcal{L}(a)$ such that there is no concept description $-\forall r.C \supseteq D$ in $\mathcal{L}(a)$, i.e. $-\forall r.C \notin \mathcal{L}^{\dagger}(a)$, where $\mathcal{L}^{\dagger}(a) :=$

$$\mathcal{L}^{+}(a) \cup \{-Qr.C \mid \{-Qr.C, +Qr.C \supseteq D\} \subseteq \mathcal{L}(a) \text{ and } +D \notin \mathcal{L}(a) \text{ for some } D\}.$$

This application is safe, because $-\forall r.C$ cannot appear in this case in $\mathcal{L}(b)$, and so it does not lead to a loop. The second rule, $(\forall d)$, applies to several (not necessarily to all) role restrictions $\{-\forall r_0.C_0, \ldots, -\forall r_n.C_n\} \subseteq \mathcal{L}^{\dagger}(a)$. This rule creates an $\preccurlyeq$-successor b of a with the labelling $\mathcal{L}^{\dagger}(a)$ and $n+1$ R-successors of b marked $r_0, \ldots, r_n$, respectively. This rule describes the situation when all duplicated concept descriptions of $\mathcal{L}(a)$ are expanded at the $\preccurlyeq$-successor b in the same way, as it was done at a. We need one more rule, viz. $(\forall s)$, which is a combination of the previous rules $(\forall-)$ and $(\supseteq Qs)$ and which corresponds to the situation when one of the concept descriptions duplicated at a becomes inactive at the $\preccurlyeq$-successor. We do not assume that $\forall r_0.C_0 \neq Qr_1.C_1$. To avoid infinite loops we must guarantee that the application of $(\forall d)$ cannot be repeated. To this end we assume that the vertex b created by the rule $(\forall d)$ is marked by "d" and the rule $(\forall d)$ cannot be applied to a vertex marked by "d".

The priority of rules will be slightly changed:

(1) $(\sqcap+)$, $(\sqcap-)$, $(\sqcup+)$, $(\sqcup-)$, $(\supseteq N)$, $(\supseteq \sqcup)$, $(\supseteq \sqcap)$, $(\supseteq\supseteq s)$, $(\supseteq \forall s)$, $(\supseteq \exists s)$;
(2) $(\supseteq \forall a)$, $(\supseteq \exists a)$;
(3) $(\preccurlyeq^{-1} r)$;
(4) $(\exists+)$, $(\exists-)$, $(\forall+)$;
(5) $(\forall-)$, $(\forall d)$, $(\forall s)$, $(\supseteq\supseteq a)$, $(\supseteq -)$.

As compared to the previous version of the algorithm, the rules of the highest priority extending the node labellings are divided into two groups. This is connected with using the Dershowitz–Manna preordering for proving termination of the algorithm. First we apply rules which do not violate the ascending of this ordering, and than the rules $(\supseteq \forall a)$ and $(\supseteq \exists a)$.

We have thus defined a modified tableau algorithm which consists of the expansion rules of Table 2 except for $(\forall-)$, the rules of Table 3, the new priority of rules and the agreement that every vertex created by rule $(\forall d)$ is marked by "d" and rule $(\forall d)$ cannot be applied to a vertex marked by "d".

Proposition 14. *The modified tableau algorithm for $\mathcal{CALC}^{\mathsf{C}}$ terminates for any concept $-D$.*

Proof. If a completion tree contains a clash it is finite, because the tableau algorithm terminates just after arriving at a clash. Assume that the tableau algorithm applied to $\{-D\}$ produced a clash-free completion tree. We prove that it is

Table 3
Modified role expansion rules for $\mathcal{CALC}^{\mathsf{C}}$

$(\forall-)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-\forall r.C\},\ -\forall r.C\notin\mathcal{L}^{\dagger}(a),\ \neg\exists b,c(a\preccurlyeq b,\ brc,\ -C\in\mathcal{L}(c))}{a\preccurlyeq b^*,\ b^*rc^*,\ \mathcal{L}(a)=\mathcal{L}'\cup\{-\forall r.C\downarrow\},\ \mathcal{L}(b^*):=(\mathcal{L}(a))^{+},\ \mathcal{L}(c^*):=\{-C\}}$
$(\forall d)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-\forall r_0.C_0,\ldots,-\forall r_n.C_n\},\ -\forall r_i.C_i\in\mathcal{L}^{\dagger}(a),\ \neg\exists b_i,c_i(a\preccurlyeq b_i,\ b_irc_i,-C_i\in\mathcal{L}(c_i)),\ 0\leqslant i\leqslant n}{a\preccurlyeq b^*,\ b^*r_ic_i^*,\ \mathcal{L}(a):=\mathcal{L}'\cup\{-\forall r_0.C_0\downarrow,\ldots,-\forall r_n.C_n\downarrow\},\ \mathcal{L}(b^*):=\mathcal{L}^{\dagger}(a),\ \mathcal{L}(c_i^*):=\{-C_i\}}$
$(\forall s)$	$\dfrac{\mathcal{L}(a)=\mathcal{L}'\cup\{-\forall r_0.C_0,+Qr_1.C_1\supseteq D\},\ Q\in\{\forall,\exists\},\ \neg\exists b,c(a\preccurlyeq b,\ br_0c,-C_0\in\mathcal{L}(c))}{a\preccurlyeq b^*,\ b^*r_0c^*,\ \mathcal{L}(b^*):=(\mathcal{L}(a))^{+}\cup\{+D,-\forall r_0.C_0\downarrow,+Qr_1.C_1\supseteq D\downarrow\},\ \mathcal{L}(c^*):=\{-C_0\}}$

finite. Now we define the weight $W(C)$ of a concept C as follows.

$$W(A) = 1, \quad \text{where } A \in N_C;$$
$$W(C \sqcup D) = W(C \supseteq D) = W(C) + W(D) + 1;$$
$$W(C \sqcap D) = W(C) + W(D) + 2;$$
$$W(Qr.C) = W(C) + 1, \quad \text{where } Q \in \{\forall, \exists\}.$$

For signed concepts we put

$$W(\epsilon C) := W(C), \quad \text{where } \epsilon \in \{+, -\}.$$

As in Proposition 12 we define the preordering of concepts wrt their weights and the respective Dershowitz–Manna preordering $\leqslant_{DM}$.

According to Proposition 12, for every node a the application of rules of priority 1 terminates. Denote by $\mathcal{L}^*(a)$ the set of active concepts in the labelling of a at a stage when all rules of priority 1 have been applied to this node. It can easily be seen that if b is a successor of a and it is not a $\preccurlyeq$-successor of a obtained by the rule $(\forall d)$ or it is not created by the rule $(\preccurlyeq^{-1} r)$, then $\mathcal{L}^*(b)$ is strictly smaller wrt $\leqslant_{DM}$ than $\mathcal{L}^*(a)$. If b is a $\preccurlyeq$-successor created by the rule $(\forall d)$ then $\mathcal{L}(b) = \mathcal{L}(a)$. Since rule $(\forall d)$ cannot be applied twice, for any successor c of b the labelling $\mathcal{L}^*(c)$ is strictly smaller wrt $\leqslant_{DM}$ than $\mathcal{L}^*(a)$. If b is obtained by $(\preccurlyeq^{-1} r)$ from the nodes a, c and d such that $a \preccurlyeq d$, arc, drb and $c \preccurlyeq b$, then it can be easily seen that $\mathcal{L}^*(b)$ is strictly smaller wrt $\leqslant_{DM}$ than $\mathcal{L}^*(a)$.

Thus the proof follows from the facts that the Dershowitz–Manna preordering is well-founded and that each node has only finitely many successors. □

Proposition 15 *(Soundness of the modified algorithm). If the modified tableau algorithm can be applied to $\{-D\}$ so that it yields a complete and clash-free completion tree, then there exists a tableau for $\{-D(a)\}$, where a is the root of the completion tree.*

Proof. A tableau can be constructed from a completion tree in exactly the same way as for the previous version of the tableau algorithm. The proof that this is really a tableau is exactly the same as in Proposition 11. □

Proposition 16 *(Completeness). If there exists a tableau for $\{-D(a)\}$, then the expansion rules can be applied in such a way that the tableau algorithm outputs a complete and clash-free completion tree for $\{-D\}$.*

Proof. We prove this proposition in the same way as Proposition 13, only at one item the definition of function π must be modified. We will say that at some stage of construction of the tree $\mathbf{T}$ a concept description $-\forall r.C \in \mathcal{L}(x)$ *requires attention* if there are no y and z such that $x \preccurlyeq y$, yrz and $-C \in \mathcal{L}(z)$.

Let $\pi(x) = b$ be already defined and let some $-\forall r_0.C_0 \in \mathcal{L}(x)$ require attention. By definition of tableaux, there are c and d such that $b \preccurlyeq c$, cr_0d and $-C_0 \in \mathcal{L}^*(d)$. If $-\forall r_0.C_0 \notin \mathcal{L}^\dagger(x)$, we apply the rule $(\forall-)$ to $-\forall r_0.C_0$. Assume $-\forall r_0.C_0 \in \mathcal{L}^\dagger(x)$, but for some $+Qr_1.C_1 \supseteq D \in \mathcal{L}(x)$ we have $\{+D, +Qr_1.C_1 \supseteq D\} \subseteq \mathcal{L}^*(c)$. In this case we apply the rule $(\forall s)$ to $-\forall r_0.C_0$ and $+Qr_1.C_1 \supseteq D$. In both cases we generate the new vertices y and z with $x \preccurlyeq y$ and yr_0z, and put $\pi(y) = c$ and $\pi(z) = d$.

If $-\forall r_0.C_0 \in \mathcal{L}^\dagger(x)$ and there are no c and d such that $b \preccurlyeq c$, cr_0d, $-C_0 \in \mathcal{L}^*(d)$, and $\{+D, +Qr_1.C_1 \supseteq D\} \subseteq \mathcal{L}^*(c)$ for some $+Qr_1.C_1 \supseteq D \in \mathcal{L}(x)$ we act as follows.

Let $-\forall r_0.C_0, -\forall r_1.C_1, \ldots, -\forall r_n.C_n$ be all negatively signed universal role restrictions from $\mathcal{L}^\dagger(x)$. Choose some c and d_0 in $\mathbf{S}$ such that $b \preccurlyeq c$, cr_0d_0 and $-C_0 \in \mathcal{L}^*(d_0)$, generate the new vertices y and z_0 with $x \preccurlyeq y$ and yr_0z_0, and put $\pi(y) = c$, $\pi(z_0) = d_0$, $\mathcal{L}(y) := \mathcal{L}^\dagger(x)$, and $\mathcal{L}(z_0) = \{-C_0\}$.

Further, we consider $-\forall r_1.C_1$. If there are c_1 and d_1 such that $c \preccurlyeq c_1$, $c_1r_1d_1$, $-C_1 \in \mathcal{L}^*(d_1)$, and for some $+Qr.C \supseteq D \in \mathcal{L}(x)$ we have $\{+D, +Qr.C \supseteq D\} \subseteq \mathcal{L}^*(c_1)$, then we apply the rule $(\forall s)$ to $-\forall r_1.C_1$ and $+Qr.C \supseteq D$. We generate the new vertices y_1 and z_1 with $y \preccurlyeq y_1$ and $y_1r_1z_1$, and put $\pi(y_1) = c_1$ and $\pi(z_1) = d_1$. Otherwise, choose some c_1 and d_1 in $\mathbf{S}$ such that $c \preccurlyeq c_1$, $c_1r_1d_1$ and $-C_1 \in \mathcal{L}^*(d_1)$, generate the new vertex z_1 with yr_1z_1, and put $\pi(z_1) = d_1$ and $\mathcal{L}(z_1) = \{-C_1\}$. Now we can consider the creation of the vertices y, z_0 and z_1 as a unique application of the rule $(\forall d)$.

Continue the analysis of the set $\mathcal{L}^{\dagger}(x)$ looking for the possible application of the rule $(\forall s)$ or for extension of the number of concept descriptions involved in the application of the rule $(\forall d)$.

Note that if the rule $(\forall d)$ was applied to all concepts $-\forall r_0.C_0, -\forall r_1.C_1, \ldots, -\forall r_n.C_n$, then no concepts of the form $-\forall r.C$ require attention in $\mathcal{L}(y)$. □

From the last two propositions we obtain

Theorem 17. *A concept description D is $\mathcal{CALC}^{\mathsf{C}}$-valid iff the expansions rules can not be applied to $\{-D\}$ so that they yield a complete and clash-free completion tree.*

Recall that the subsumption $C \sqsubseteq D$ is equivalent to the validity of $C \sqsupseteq D$.

Corollary 18. *Let C and D be concept descriptions in nnf. The concept C is subsumed by D iff the expansions rules can not be applied to $\{-(C \sqsupseteq D)\}$ so that they yield a complete and clash-free completion tree.*

In conclusion, we show how the described tableau algorithm can be adopted to work with ABoxes. In this case the algorithm will construct not a completion tree, but a completion graph. We call a signed ABox $\mathcal{A}$ *clash-free* if it does not contain a subset of the form $\{+C(a), -C(a)\}$ or $\{+r(a,b), -r(a,b)\}$. Let $\mathcal{A}$ be a clash-free signed ABox. With $\mathcal{A}$ we associate a graph $\mathbf{G}_{\mathcal{A}}$ with set of nodes $\{a \in N_I \mid a \text{ occur in } \mathcal{A}\}$, set of edges $\{(a,b) \mid +r(a,b) \in \mathcal{A}\}$ and labelling $\mathcal{L}$ of nodes and edges defined as follows: $\mathcal{L}(a) := \{\epsilon B \mid \epsilon B(a) \in \mathcal{A}\}$, $\epsilon \in \{+,-\}$ and $\mathcal{L}((a,b)) = \{r \mid +r(a,b) \in \mathcal{A}\}$. This graph can be expanded with the help of expansion rules of the modified tableau algorithm in the same way as it was done earlier for a completion tree, and we construct in this way a completion graph. The notions of clash and completeness are exactly the same as for a completion tree.

Similarly to Proposition 14 one can prove.

Proposition 19. *Let $\mathcal{A}$ be a clash-free signed ABox. The modified tableau algorithm for $\mathcal{CALC}^{\mathsf{C}}$ terminates for the graph $\mathbf{G}_{\mathcal{A}}$.*

Theorem 17 can be modified in the following way.

Theorem 20. *Let $\mathcal{A}$ be an ABox and $C(a)$ an ABox statement, $C(a) \notin \mathcal{A}$. Assume that C and all statements of $\mathcal{A}$ are in negation normal form. Put $\mathcal{A}^{\epsilon} := \{+\alpha \mid \alpha \in \mathcal{A}\} \cup \{-C(a)\}$. We have $\mathcal{A} \models C(a)$ iff the expansion rules cannot be applied to $G_{\mathcal{A}^{\epsilon}}$ so that they yield a complete and clash-free completion graph.*

6. Conclusion

To conclude this article, we discuss some directions of further investigations. First of all, we did not study here complexity questions. A very rough estimation of the complexity of the described tableau algorithm can be obtained as follows. Our construction essentially is based on [5]. The calculus for intuitionistic propositional logic suggested in this work can be considered as a propositional version of our tableau calculus. It is known that the calculus from [5] has chains of reductions which in the worst case are exponential in the size of the refuted formula. The same estimation remains obviously true for our algorithm. Thus, we obtain as an estimation of complexity 2-*NEXPTIME*. The tableau algorithm described in [8] has the same estimation of complexity, and, in fact, both algorithms work with similar objects. In [8], there were considered transitive role relations, and we interpreted intuitionistic implication via a transitive role interacting in a special way with other role relations. The algorithm of [8] was effectively realized. Due to the mentioned similarity of the algorithms, our algorithm for $\mathcal{CALC}^{\mathsf{C}}$ at least potentially can be considered as a basis for implementation. In any case, the presented algorithm for $\mathcal{CALC}^{\mathsf{C}}$ is the first example of an elementary decision procedure for a constructive description logic.

The calculus of [5], which forms the propositional basis of our tableau algorithm, was modified by J. Hudelmaier [9]. As a result, Hudelmaier obtained an $O(n \log n)$-space decision procedure for propositional intuitionistic logic. If Hudelmaier's ideas could be combined with our algorithm, one can essentially decrease the complexity of the decision procedure for $\mathcal{CALC}^{\mathsf{C}}$.

Another possible direction of work is as follows. We noted above that the connective $\supseteq$ in the language of $\mathcal{CALC}^{\mathsf{C}}$ expresses the subsumption relation. So it is reasonable to look for applications of our algorithm to work with TBoxes.

The next topic is to increase the expressive power of constructive description logic. In this article we presented a constructive analogue only of the simplest description logic $\mathcal{ALC}$. The presented tableau algorithm allows it to make the observation that in the construction of a completion tree, extending R-subtrees and extending $\preccurlyeq$-subtrees are rather independent of each other. One can say that the work with operators of classical description logic and the work with constructive implication go in orthogonal directions. So, one can expect that the expressive power of $\mathcal{CALC}^{\mathsf{C}}$ can be enlarged along familiar lines. We did not try to do it in this work, because our goal was to give a model case of a tableau algorithm for constructive description logic and to look for proof-theoretical ideas suitable for such algorithms.

References

[1] A.R. Anderson, N.D. Belnap, Entailment: The Logic of Relevance and Necessity, vol. I, Princeton University Press, Princeton, 1975.
[2] N.D. Belnap, A useful four-valued logic, in: J.M. Dunn, G. Epstein (Eds.), Modern Uses of Multiple-Valued Logic, Reidel, Dordrecht, 1977, pp. 8–37.
[3] J.M. Dunn, Intuitive semantics for first-degree entailment and ‘coupled trees’, Philosophical Studies 29 (1976) 149–168.
[4] J.M. Dunn, Partiality and its dual, Studia Logica 66 (2000) 5–40.
[5] R. Dyckhoff, Contraction-free sequent calculi for intuitionistic logic, Journal of Symbolic Logic 57 (1992) 795–807.
[6] G. Fischer Servi, Axiomatizations for some intuitionistic modal logics, Red. Sem. Mat. Universi. Politec. Torino 42 (1984) 179–194.
[7] C. Grefe. Fischer Servi’s intuitionistic modal logic has the finite model property, in: M. Kracht, et al. (Eds.), Advances in Modal Logic, vol. 1, CSLI Lecture Notes, vol. 87, 1998, pp. 85–98.
[8] I. Horrocks, U. Sattler, A description logic with transitive and inverse roles and role hierarchies, Journal of Logic and Computation 9 (1999) 385–410.
[9] J. Hudelmaier, An $\mathrm{O}(n \log n)$-space decision procedure for intuitionistic propositional logic, Journal of Logic and Computation 3 (1993) 63–75.
[10] N. Kamide, Phase semantics and Petri net interpretations for resource-sensitive strong negation, Journal of Logic, Language and Information 15 (2006) 371–401.
[11] C. Meghini, U. Straccia, A relevance terminological logic for information retrieval, in: Proceedings of SIGIR-96, 19th International Conference on Research and Development in Information Retrieval, Zurich, Switzerland, 1996, pp. 197–205.
[12] S.P. Odintsov, H. Wansing, Inconsistency-tolerant description logic: Motivation and basic systems, in: V.F. Hendricks, J. Malinowski (Eds.), Trends in Logic: 50 Years of Studia Logica, Kluwer Academic Publishers, Dordrecht, 2003, pp. 301–335.
[13] S.P. Odintsov, H. Wansing, Constructive predicate logic and constructive modal logic. Formal duality versus semantical duality, in: V. Hendricks, et al. (Eds.), First-Order Logic Revisited, Logos Verlag, Berlin, 2004, pp. 269–286.
[14] V. de Paiva, Constructive description logics: What, why and how, extended draft, Presented at “Context Representation and Reasoning”, Riva del Garda, August 2006, available at: http://www.cs.bham.ac.uk/%7Evdp/publications/papers.html.
[15] P.F. Patel-Schneider, A four-valued semantics for terminological logics, Artificial Intelligence 38 (3) (1989) 319–351.
[16] G. Priest, An Introduction to Non-Classical Logic, Cambridge UP, Cambridge, 2001.
[17] Y. Shramko, H. Wansing, Some useful sixteen-valued logics: How a computer network should think, Journal of Philosophical Logic 34 (2005) 121–153.
[18] A.K. Simpson, The proof theory and semantics of intuitionistic modal logics, PhD Thesis, University of Edinburgh, 1994.
[19] U. Straccia, A sequent calculus for reasoning in four-valued description logics, in: Proc. of the Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX-97), in: LNAI, vol. 1227, Springer, Berlin, 1997, pp. 343–357.
[20] H. Wansing, The Logic of Information Structures, LNAI, vol. 681, Springer, Berlin, 1993.