Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Towards a Tableau Algorithm for Fuzzy $\mathcal{ALC}$ with Product T-norm

Rafael Peñaloza

LTCS-Report 11-03

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Towards a Tableau Algorithm for Fuzzy $\mathcal{ALC}$ with Product T-norm

Rafael Peñaloza

Abstract

Very recently, the tableau-based algorithm for deciding consistency of general fuzzy DL ontologies over the product t-norm was shown to be incorrect, due to a very weak blocking condition. In this report we take the first steps towards a correct algorithm by modifying the blocking condition, such that the (finite) structure obtained through the algorithm uniquely describes an infinite system of quadratic constraints. We show that this procedure terminates, and is sound and complete in the sense that the input is consistent iff the corresponding infinite system of constraints is satisfiable.

1 Introduction

The panorama of fuzzy Description Logics (fuzzy DLs) changed drastically when it was shown that the basic logic $\mathcal{ALC}$ with t-norm semantics lacks the finite model property [4, 5]. This result in particular invalidated the proof of correctness of the algorithm presented in [6] for an expressive fuzzy DL over the product t-norm, and triggered a series of negative results. It was first shown independently in [5] and [1] that the algorithm from [6] is unable to detect some inconsistent ontologies. Then, ontology consistency has been shown to be undecidable for several variants of fuzzy DLs and semantics [1, 2, 3, 8, 10]. However, none of these undecidability results corresponds precisely to the logic considered in [6].

The algorithm from [6] uses a tableau-like procedure to construct a set of quadratic constraints, that then is solved by an external constraint solver. Since the algorithm needs to deal with general ontologies, the standard tableau algorithm may not terminate. This problem arises already for crisp description logics, and is solved in the DL literature through a cycle-checking technique usually called *blocking*. The idea behind blocking is to stop the execution of the algorithm once enough information for generating a model has been produced. For crisp $\mathcal{ALC}$, blocking is triggered once two equivalent nodes–that is, nodes containing the same concepts–are found. In [6], this blocking condition is used also for fuzzy $\mathcal{ALC}$, without any consideration of the fuzzy degree with which the concepts are

satisfied. However, as the examples in [5, 1] show, this blocking condition is too weak, and hence the algorithm stops before the inconsistency of the ontology is detected. A simple idea for strengthening the blocking condition is to ensure that the fuzzy degrees of all concepts are also equivalent in both, the blocking and the blocked node. Unfortunately, since this logic does not have the finite model property, such a condition may never trigger, yielding a non-terminating procedure.

In this report we introduce a new tableau-based algorithm, based on the principal ideas of [6], and improved with a new blocking condition that yields a sound, complete and terminating reduction from the problem of ontology consistency of fuzzy $\mathcal{ALC}$ with product t-norm to satisfiability of a system of quadratic constraints. This, however, does not show decidability for this logic, as the algorithm outputs a finite representation of an infinite system of constraints, for which, to the best of our knowledge, no decidability results are yet known.

The work is divided as follows. We first introduce the syntax and semantics of the fuzzy DL under consideration. Following [9], we call this logic Π-$\mathcal{ALC}$. Afterwards, we introduce the algorithm producing an infinite system of constraints. We show that this algorithm is correct and terminates.

2 The Logic Π-$\mathcal{ALC}$

The syntax of Π-$\mathcal{ALC}$ is the same as for crisp $\mathcal{ALC}$. From two disjoint sets $\mathcal{N}_\mathcal{C}$ and $\mathcal{N}_\mathcal{R}$ of *concept-* and *role-names*, respectively, Π-$\mathcal{ALC}$ concepts are built through the syntactic rule:

$$C ::= \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \exists r.C \mid \forall r.C,$$

where $A \in \mathcal{N}_\mathcal{C}$ and $r \in \mathcal{N}_\mathcal{R}$.

The syntax of the axioms in this logic is slightly different from the crisp case since they also include a bound to the *degree of truth* with which they must hold. A Π-$\mathcal{ALC}$ *ABox* is a set of *assertion axioms* of the form $\langle a : C, q\rangle$ or $\langle (a, b) : r, q\rangle$, where C is a Π-$\mathcal{ALC}$ concept, $r \in \mathcal{N}_\mathcal{R}$, q is a rational number in $[0, 1]$, and $a, b \in \mathcal{N}_\mathcal{I}$, with $\mathcal{N}_\mathcal{I}$ the set of *individual names*. A Π-$\mathcal{ALC}$ *TBox* is a set of *concept inclusion axioms* of the form $\langle C \sqsubseteq D, q\rangle$, where C, D are Π-$\mathcal{ALC}$ concepts and q is a rational in $[0, 1]$. A *fuzzy ontology* is the union of a Π-$\mathcal{ALC}$ ABox and a Π-$\mathcal{ALC}$ TBox.

The semantics of this logic extend the classical semantics of $\mathcal{ALC}$ by interpreting concept and roles as fuzzy sets over an interpretation domain. Given a domain Δ, a *fuzzy set* is a function $F : \Delta \to [0, 1]$, with the intuition that an element $\delta \in \Delta$ belongs to F with *degree* $F(\delta)$. The semantics of the different concept constructors depend on the class of fuzzy operators chosen. In the logic Π-$\mathcal{ALC}$,

we use the binary operators *product t-norm* $\otimes$, *product t-conorm* $\oplus$ and *residuum* $\rightarrow$ over the interval $[0,1]$. These operators are defined as follows, for every $\alpha, \beta \in [0,1]$:

$$\begin{aligned} \alpha \otimes \beta &:= \alpha \cdot \beta, \\ \alpha \oplus \beta &:= \alpha + \beta - \alpha \cdot \beta, \\ \alpha \rightarrow \beta &:= \begin{cases} 1 & \text{if } \alpha \leq \beta \\ \beta/\alpha & \text{otherwise.} \end{cases} \end{aligned}$$

It is worth noticing that for every $\alpha, \beta, q \in [0,1]$ it holds that $\alpha \rightarrow \beta \geq q$ iff $\beta \geq q \cdot \alpha$. In particular this means that $\alpha \rightarrow \beta \geq 1$ iff $\beta \geq \alpha$. We will make use of this property when describing the reasoning algorithms.

Definition 1 (semantics of Π-$\mathcal{ALC}$)**.** An *interpretation* is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain*, and the function $\cdot^{\mathcal{I}}$ maps each individual name a to an element of $\Delta^{\mathcal{I}}$, each concept name A to a function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0,1]$ and each role name r to a function $r^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0,1]$. The interpretation function is extended to arbitrary Π-$\mathcal{ALC}$ concepts as follows. For every $\delta \in \Delta^{\mathcal{I}}$,

$$\begin{aligned} \top^{\mathcal{I}}(\delta) &= 1 \\ \bot^{\mathcal{I}}(\delta) &= 0 \\ (C_1 \sqcap C_2)^{\mathcal{I}}(\delta) &= C_1^{\mathcal{I}}(\delta) \cdot C_2^{\mathcal{I}}(\delta) \\ (C_1 \sqcup C_2)^{\mathcal{I}}(\delta) &= C_1^{\mathcal{I}}(\delta) + C_2^{\mathcal{I}}(\delta) - C_1^{\mathcal{I}}(\delta) \cdot C_2^{\mathcal{I}}(\delta) \\ (\neg C)^{\mathcal{I}}(\delta) &= 1 - C^{\mathcal{I}}(\delta) \\ (\exists r.C)^{\mathcal{I}}(\delta) &= \sup_{\gamma \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(\delta, \gamma) \cdot C^{\mathcal{I}}(\gamma) \\ (\forall r.C)^{\mathcal{I}}(\delta) &= \inf_{\gamma \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(\delta, \gamma) \rightarrow C^{\mathcal{I}}(\gamma). \end{aligned}$$

Notice that, contrary to the crisp case, existential and universal restrictions are not dual to each other; that is, in general $(\neg \exists r.C)^{\mathcal{I}}(\delta) \neq (\forall r.\neg C)^{\mathcal{I}}(\delta)$.

Definition 2 (model)**.** An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies the axiom $\langle a : C, q \rangle$ iff $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq q$, it satisfies $\langle (a,b) : r, q \rangle$ iff $r^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq q$ and it satisfies the concept inclusion $\langle C \sqsubseteq D, q \rangle$ iff $\inf_{\delta \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(\delta) \rightarrow D^{\mathcal{I}}(\delta) \geq q$. This interpretation is called a *model* of the ontology $\mathcal{O}$ if it satisfies all the axioms in $\mathcal{O}$.

A model is called *witnessed* if for every $\delta \in \Delta^{\mathcal{I}}$, role r and concept C there exist $\gamma, \gamma' \in \Delta^{\mathcal{I}}$ such that

- $(\exists r.C)^{\mathcal{I}}(\delta) = r^{\mathcal{I}}(\delta, \gamma) \cdot C^{\mathcal{I}}(\gamma)$, and
- $(\forall r.C)^{\mathcal{I}}(\delta) = r^{\mathcal{I}}(\delta, \gamma') \rightarrow C^{\mathcal{I}}(\gamma')$.

Although it has been shown in [7] that fuzzy $\mathcal{ALC}$ does not have the witnessed model property, reasoning is sometimes restricted to witnessed models only (see, for instance [6, 7]). The restriction to witnessed models is useful when trying to design a reasoning algorithm that deals locally with every individual created. Indeed, as the semantics of the existential restriction $\exists r.C$ consider a supremum over all individuals of the domain, it is possible that such supremum is not reached by any individual of the (infinite) domain. An algorithm for general models would then have to be able to deal with such a "global" condition when deciding consistency. The restriction to witnessed models allows the algorithm to change this condition to a local one by producing one individual that is the witness for this supremum, and hence changing it to a maximum. Since our tableaux algorithm will have only local rules, it will deal only with witnessed models.

3 The Algorithm

A tableau-based algorithm for deciding consistency of a fuzzy ontology in Π-$\mathcal{ALCF}(D)$ was presented in [6].[1] The algorithm applies rules to produce a system of quadratic constraints that verify that the fuzzy semantics are satisfied by a tree-like interpretation. However, this algorithm has been shown to be incorrect, even if restricted to fuzzy $\mathcal{ALC}$ [5, 1], since it uses a very weak blocking condition. We will now describe a modified algorithm that fixes the problems of [6] by providing a stronger blocking condition, such that the cyclic structure recognized by it uniquely determines an infinite system of quadratic constraints. The tableau-based algorithm obtained this way is ensured to terminate, and is sound and complete in the sense that the input is consistent iff the corresponding infinite constraint system is satisfiable.

In the following, x (possibly with sub- or superindices) denotes a *continuous variable* taking values from $[0, 1]$, q denotes a *constant* in $[0, 1]$, l denotes a *literal*, i.e., a continuous variable x, a *negated variable* $1 - x$ or a constant, and y denotes a *Boolean variable* taking values from $\{0, 1\}$.

The algorithm constructs a completion forest; one can think of this as a collection of trees whose roots may be arbitrarily interconnected by edges. Every node v in this forest is labeled by a set $\mathcal{L}(v)$ of labeled concepts of the form $\langle C, l\rangle$ and a set $\mathcal{C}(v)$ of constraints. Intuitively, $\langle C, l\rangle \in \mathcal{L}(v)$ means that v belongs to C with degree at least l and the constraints in $\mathcal{C}(v)$ restrict the valuations of the different variables used. Additionally, every edge (v_1, v_2) is labeled with a set $\mathcal{L}(v_1, v_2)$ of labeled role names $\langle r, l\rangle$ with the meaning that (v_1, v_2) are in an r-relation with degree at least l.

The algorithm initializes the completion forest to contain one *root node* v_i for each

[1] Π-$\mathcal{ALCF}(D)$ extends Π-$\mathcal{ALC}$ with additional constructors that are not relevant for this report.

(A)	if $\langle A, l\rangle \in \mathcal{L}(v)$ then add $x_{v:A} \geq l$ to $\mathcal{C}(v)$
$(\overline{A})$	if $\langle \neg A, l\rangle \in \mathcal{L}(v)$ then add $x_{v:A} \leq 1 - l$ to $\mathcal{C}(v)$
(r)	if $\langle r, l\rangle \in \mathcal{L}(v, w)$ then add $x_{(v,w):r} \geq l$ to $\mathcal{C}(w)$
$(\top)$	if $\langle \top, l\rangle \in \mathcal{L}(v)$ or $\langle \neg\bot, l\rangle \in \mathcal{L}(v)$ then add $l = 1$ to $\mathcal{C}(v)$
$(\bot)$	if $\langle \bot, l\rangle \in \mathcal{L}(v)$ or $\langle \neg\top, l\rangle \in \mathcal{L}(v)$ then add $l = 0$ to $\mathcal{C}(v)$
$(\neg\neg)$	if $\langle \neg\neg C, l\rangle \in \mathcal{L}(v)$ then add $\langle C, l\rangle$ to $\mathcal{L}(v)$
$(\sqcap)$	if $\langle C \sqcap D, l\rangle \in \mathcal{L}(v)$ then add $\langle C, x_1\rangle, \langle D, x_2\rangle$ to $\mathcal{L}(v)$ and $x_1 \cdot x_2 \geq l$ to $\mathcal{C}(v)$
$(\overline{\sqcap})$	if $\langle \neg(C \sqcap D), l\rangle \in \mathcal{L}(v)$ then add $\langle \neg C \sqcup \neg D, l\rangle$ to $\mathcal{L}(v)$
$(\sqcup)$	if $\langle C \sqcup D, l\rangle \in \mathcal{L}(v)$ then add $\langle C, x_1\rangle, \langle D, x_2\rangle$ to $\mathcal{L}(v)$ and $x_1 + x_2 - x_1 \cdot x_2 \geq l$ to $\mathcal{C}(v)$
$(\overline{\sqcup})$	if $\langle \neg(C \sqcup D), l\rangle \in \mathcal{L}(v)$ then add $\langle \neg C \sqcap \neg D, l\rangle$ to $\mathcal{L}(v)$
$(\forall)$	if $\langle \forall r.C, l_1\rangle \in \mathcal{L}(v)$ and $\langle r, l_2\rangle \in \mathcal{L}(v, w)$ then add $\langle C, x\rangle$ to $\mathcal{L}(w)$ and $x \geq l_1 \cdot l_2$ to $\mathcal{C}(w)$
$(\overline{\exists})$	if $\langle \neg\exists r.C, l_1\rangle \in \mathcal{L}(v)$ and $\langle r, l_2\rangle \in \mathcal{L}(v, w)$ then add $\langle \neg C, 1 - x_1\rangle$ to $\mathcal{L}(w)$ and $x_{(v,w):r} \leq x_2, x_1 \cdot x_2 \leq 1 - l_1$ to $\mathcal{C}(w)$
$(\overline{\forall})$	if $\langle \neg\forall r.C, l\rangle \in \mathcal{L}(v)$ then create a new node w and add $\langle r, x_1\rangle$ to $\mathcal{L}(v, w)$, $\langle C, x_2\rangle$ to $\mathcal{L}(w)$ and $y \cdot x_1 + (1 - y) \cdot x_2 \geq y \cdot x_2 + (1 - y) \cdot x_1, l \leq y, y \cdot x_2 \leq x_1 - l \cdot x_1$ to $\mathcal{C}(w)$
$(\exists)$	if $\langle \exists r.C, l\rangle \in \mathcal{L}(v)$ then create a new node w and add $\langle r, x_1\rangle$ to $\mathcal{L}(v, w)$, $\langle C, x_2\rangle$ to $\mathcal{L}(w)$ and $x_1 \cdot x_2 \geq l$ to $\mathcal{C}(w)$
$(\sqsubseteq)$	if $\langle C \sqsubseteq D, q\rangle \in \mathcal{T}$ and v is a node in the forest, then add $\langle \neg C, 1 - x_1\rangle, \langle D, x_2\rangle$ to $\mathcal{L}(v)$ and $x_2 \geq x_1 \cdot q$ to $\mathcal{C}(v)$.

Table 1: Completion rules for fuzzy $\mathcal{ALC}$ consistency. x, x_1, x_2, y are new variables

individual name a_i appearing in the ABox $\mathcal{A}$ with $\mathcal{L}(v_i) = \{\langle C, l\rangle \mid \langle a_i : C, l\rangle \in \mathcal{A}\}$ and $\mathcal{L}(v_i, v_j) = \{\langle r, l\rangle \mid \langle (a_i, a_j) : r, l\rangle \in \mathcal{A}\}$. The sets $\mathcal{C}(v_i)$ are all initialized as empty. This completion forest is extended by application of the completion rules from Table 1. As is standard with tableau-based algorithms, these rules are only applied as long as something new is added to the completion forest.

The main idea of these rules is that they decompose complex concepts into their subconcepts, while preserving the fuzzy semantics through the restrictions in $\mathcal{C} := \bigcup_v \mathcal{C}(v)$. The initialization and rule applications are consistency preserving.

Lemma 3. *If an ontology $\mathcal{O}$ is witnessed consistent, then after every rule application the system of contraints $\mathcal{C}$ is satisfiable.*

Proof. Since $\mathcal{O}$ is consistent, there is a model $\mathcal{I}$ of $\mathcal{O}$. We will show by induction on the rule applications that this model is a solution for the system $\mathcal{C}$ at every

step. To do this, we will build a function f from the nodes of the completion forest to $\Delta^{\mathcal{I}}$ such that for every node v and every $\langle C, \ell\rangle \in \mathcal{L}(v)$ (resp. every $\langle r, \ell\rangle \in \mathcal{L}(v, v')$) it holds that $C^{\mathcal{I}}(f(v)) \geq \widehat{\ell}$ (resp. $r^{\mathcal{I}}(f(v), f(v')) \geq \widehat{\ell}$), where $\widehat{\ell} = \ell$ if ℓ is a constant and $\widehat{\ell}$ is the valuation of ℓ if it is a variable or a negated variable.

After the initialization, the system $\mathcal{C}$ is empty and hence is trivially satisfiable. Given a root node v, let a be the individual name in the ABox from which v was produced. We then set $f(v) = a^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $\mathcal{O}$, this function satisfies the condition described above.

Suppose now that the condition holds at some step of the execution and that a rule is applied. We show by a case analysis on the rule used that after the rule application, it still holds.

(A) The rule (A) only adds the restriction $x_{v:A} \geq \ell$ to $\mathcal{C}(v)$. By induction hypothesis, we know that $A^{\mathcal{I}}(f(v)) \geq \widehat{\ell}$ and hence setting $x_{v:A} := A^{\mathcal{I}}(f(v))$ yields a satisfying valuation of $\mathcal{C}$. The rest of the condition is untouched.

($\sqcap$) Let $\langle C \sqcap D, \ell\rangle \in \mathcal{L}(v)$. By induction hypothesis we know that $(C \sqcap D)^{\mathcal{I}}(f(v)) \geq \widehat{\ell}$. We can set the valuation $x_1 := C^{\mathcal{I}}(f(v))$ and $x_2 := D^{\mathcal{I}}(f(v))$. We thus obtain $x_1 \cdot x_2 = (C \sqcap D)^{\mathcal{I}}(f(v)) \geq \ell$ and $C^{\mathcal{I}}(f(v)) \geq x_1, D^{\mathcal{I}}(f(v)) \geq x_2$, as desired.

($\forall$) Since $(\forall r.C)^{\mathcal{I}}(f(v)) = \inf_{\delta \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(f(v), \delta) \geq \widehat{\ell_1}$ and $r^{\mathcal{I}}(f(v), f(w)) \geq \widehat{\ell_2}$ we know that $C^{\mathcal{I}}(f(w)) \geq \ell_1 \cdot \ell_2$, and hence setting $x := C^{\mathcal{I}}(f(w))$ yields the desired condition.

($\exists$) As $(\exists r.C)^{\mathcal{I}}(f(v)) \geq \widehat{\ell}$ and $\mathcal{I}$ is witnessed, we know that there must exist a node $\delta \in \Delta^{\mathcal{I}}$ such that $(\exists r.C)^{\mathcal{I}}(f(v)) = r^{\mathcal{I}}(f(v), \delta) \cdot C^{\mathcal{I}}(\delta)$. We thus define $f(w) := \delta$ and $x_1 := r^{\mathcal{I}}(f(v), \delta), x_2 := C^{\mathcal{I}}(\delta)$ to satisfy the condition.

($\sqsubseteq$) Since $\mathcal{I}$ is a model of $\mathcal{O}$, it must satisfy $C^{\mathcal{I}}(f(v)) \to D^{\mathcal{I}}(f(v)) \geq q$. We thus set $x_1 := C^{\mathcal{I}}(f(v)), x_2 := D^{\mathcal{I}}(f(v))$ to obtain the desired condition.

All other rules can be treated analogously. □

Remark. The proof depends on the first condition of witnessed models, and does not hold for general models.

Assume for the moment that the completion forest is saturated; that is, no completion rule is further applicable. This completion forest describes a model, where the membership degrees of a node v to a concept name A corresponds to the value of the variable $x_{v:A}$. Thus, if the constraint system $\mathcal{C}$ is satisfiable, then we can build a model for the ontology, which means that it is consistent.

Lemma 4. *Consider the (possibly non-terminating) algorithm consisting of a fair application of all completion rules to an ontology $\mathcal{O}$. $\mathcal{O}$ is witnessed consistent if after every rule application the system of constraints $\mathcal{C}$ is satisfiable.*

Proof. Notice first that the solution space of each constraint added to $\mathcal{C}$ by a rule application is closed and hence the solution of any finite set of such constraints is also closed. Since rule applications only add constraints, we have a sequence of decreasing closed solution spaces. Each of these solution spaces is by assumption non-empty, and hence this sequence converges to a non-empty solution set. Any element of this solution set can be used to build an interpretation $\mathcal{I}$ for $\mathcal{O}$, using as domain the infinite set of nodes generated by rule applications and $A^{\mathcal{I}}(v) = x_{v:A}$ for each node v and concept name A. We need now to show that this interpretation is indeed a model of $\mathcal{O}$.

Let v be a node in the completion forest created by this algorithm, and $\langle C, \ell\rangle \in \mathcal{L}(v)$. Since we use a fair rule application, we know that at some step, we will apply a completion rule to this labeled concept. From a simple case analysis and induction argument on the structure of C, it can be shown that $C^{\mathcal{I}}(v) \geq \widehat{\ell}$: for the base case, C is a concept name, then $C^{\mathcal{I}}(v) = x_{v:C} = \widehat{\ell}$; for complex concepts, the property follows from the decomposition rule used. In particular, this implies that, for every $\langle C \sqsubseteq D, q\rangle \in \mathcal{O}$ and every $v \in \Delta^{\mathcal{I}}$ it holds that $C^{\mathcal{I}}(v) \to D^{\mathcal{I}}(v) \geq q$, and hence $\mathcal{I}$ is a model of $\mathcal{O}$.

This model is trivially witnessed, since every node has finitely many successors with degree greater than 0. □

Hence, we have a sound and complete precedure for deciding consistency of a Π-$\mathcal{ALC}$ ontology. However, as is also the case for crisp $\mathcal{ALC}$, this completion forest construction may not terminate, due to the presence of cyclic axioms. In order to ensure termination, a blocking condition is used, which disallows the application of "generating" rules; i.e., the $(\exists)$ and $(\overline{\forall})$ rules cannot be applied in any node that is considered blocked. Intuitively, one can stop the execution of the algorithm once we can find nodes in every path that are equivalent, since we have all the information required for building the full completion forest. While a correct definition of equivalence of nodes is simple for crisp $\mathcal{ALC}$, in the case of Π-$\mathcal{ALC}$ it requires a more detailed analysis.

4 Adding Blocking

In the case of Π-$\mathcal{ALC}$, equivalence of nodes is defined in terms of the labeled concepts and the restrictions they define. The idea behind the blocking condition is that once we produce a node that is equivalent to a previously explored one, then one does not need to continue expanding the tree, since all the information

required has been already produced. Intuitively, one could produce a cycle between the blocked and the blocking node, which represents the whole model. In the case of Π-$\mathcal{ALC}$, due to the lack of the finite model property, this cycle may be more complex, since the same concepts may be considered with different truth degrees at every node. In this case, we need to check that the constraints defined by the nodes are isomorphic. Although this does not define a finite (cyclic) model, it gives a finite description of the infinite system of constraints that would be produced by the algorithm without blocking.

Definition 5 (blocking)**.** Two nodes v, w are *equivalent*, denoted as $\mathcal{L}(v) \approx \mathcal{L}(w)$, if there exists an isomorphism f between $\mathcal{C}(v)$ and $\mathcal{C}(w)$ such that for every concept C and literal ℓ in $\mathcal{C}(v)$, $\langle C, \ell\rangle \in \mathcal{L}(v)$ iff $\langle C, f(l)\rangle \in \mathcal{L}(w)$.

A node v is *directly blocked* iff it is not a root node and it has an ancestor w such that $\mathcal{L}(v) \approx \mathcal{L}(w)$; in this case we say that w is the *blocking* node of v. A node is *blocked* if it is directly blocked or its predecessor is blocked.

When a node is (directly or indirectly) blocked, then none of the rules $(\exists)$ or $(\overline{\forall})$ may be applied, disallowing this way the creation of new individuals. Notice that none of the other rules produce any new individuals or edges; they only decompose the information contained in the respective node to basic concepts. As the following lemma shows, this notion suffices for obtaining a terminating procedure.

Lemma 6. *If no generating rule is applied over blocked nodes, then the algorithm terminates.*

Proof. The algorithm starts with a graph structure that is extended in a tree-like manner. The branching of these trees is bounded by the number of existential $(\exists)$ and negated universal $(\overline{\forall})$ concepts appearing in the ontology, and hence is finite. Each node is labeled with a finite set of labeled concepts, where each concept may appear at most as many times as it occurs in the ontology and a finite set of constraints. Since every rule adds at least one such labeled concept to a node or at least one constraint, only finitely many rules applications can be triggered at any specific node.

It remains only to show that each of these trees has also finite depth; that is, that a blocked node is found eventually on each branch. Once again, every node is generated by a finite number of rule applications. Since every rule adds at most three variables to the set $\mathcal{C}$, each node has finitely many variables. Every inequality in $\mathcal{C}$ uses at most three variables. Thus, there are also finitely many sets of inequalities up to isomorphism. This means that the tree has a finite depth, since every path long enough will get a blocked node. Hence the whole tree is finite. Thus the algorithm terminates. □

When the algorithm terminates, it produces a finite forest where every leaf is blocked. We can prune this forest in such a way that every leaf is *directly* blocked.

We call this the *prunned forest*. This forest has an associated finite system of constraints. We can then "unravel" this system into an infinite one that characterizes ontology consistency.

Let $\mathcal{F}$ be a prunned forest with n leafs and block a function that maps every leaf (and hence directly blocked) node to its blocking node. For every leaf v_i, $1 \leq i \leq n$, let t_i be the subtree of t with root $\mathsf{block}(v_i)$ with nodes labeled with their respective set of restrictions $\mathcal{C}$. We call $\mathcal{C}_i$ the set of all restrictions appearing in this tree. We define the binary relation $\mapsto \subseteq \{1,\ldots,n\} \times \{1,\ldots,n\}$ where $i \mapsto j$ iff v_j is a successor of $\mathsf{block}(v_i)$. Finally, we define the language L_t as the smallest set of words in $\{1,\ldots,n\}^*$ that contains $\{\varepsilon, 1,\ldots,n\}$ and such that if $\eta i \in L_t$ and $i \mapsto j$, then also $\eta ij \in L_t$.

Definition 7. We define $\mathcal{C}_\mathcal{O}$ as the system of constraints that contains $\mathcal{C}_\varepsilon := \mathcal{C}$ and for each word ηi in L_t a disjoint copy of $\mathcal{C}_i$ and for each variable x of $\mathcal{C}(v_i)$ the restriction $x^\eta = f(x)^{\eta i}$, where f is the isomorphism between $\mathcal{C}(v_i)$ and $\mathcal{C}(\mathsf{block}(v_i))$.

Intuitively, $\mathcal{C}_\mathcal{O}$ describes how the infinite system of constraints would look like if the infinite completion forest was constructed without using the blocking condition. However, the forest itself is not constructed.

Theorem 8. *Let $\mathcal{C}_\mathcal{O}$ be the system of constraints obtained from the prunned output forest of the algorithm applied to the ontology $\mathcal{O}$. Then $\mathcal{C}_\mathcal{O}$ is satisfiable iff $\mathcal{O}$ is consistent.*

Proof. Consider the non-terminating algorithm that uses the completion rules without any blocking condition. It is easy to show by induction on rule application that at every step of this algorithm, the system of constraints $\mathcal{C}$ is contained in $\mathcal{C}_\mathcal{O}$. Thus, if $\mathcal{C}_\mathcal{O}$ is satisfiable, then at every rule application $\mathcal{C}$ is also satisfiable. From Lemma 4 it follows that $\mathcal{O}$ is consistent.

Conversely, if $\mathcal{C}_\mathcal{O}$ is not satisfiable, then by compactness of this system it follows that there must be a finite subset $\mathcal{C}$ of $\mathcal{C}_\mathcal{O}$ that is not satisfiable. Then, there is a finite number of rule applications that produce a system $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{C}$, and hence $\mathcal{C}$ is not satisfiable. Lemma 3 yields inconsistency of $\mathcal{O}$. □

5 Conclusions

We have provided the first steps towards an algorithm for deciding consistency of Π-$\mathcal{ALC}$ ontologies, by giving a sound, complete and terminating reduction to satisfiability of an infinite system of quadratic constraints. This method does not yield a decision procedure, since it is not clear how to test whether the finitely described infinite constraint system is satisfiable. The next steps in this direction

correspond to finding a bound on the number of constraints that are necessary for finding inconsistencies in the system, thus obtaining a finite system for which satisfiability can be decided.

It should be clear that the same blocking condition introduced here would yield a similar reduction for the fuzzy DL Ł-$\mathcal{ALC}$, whose semantics is based on the Łukasiewicz t-norm. The main difference is that the constraint system obtained will consist of only linear inequalities. Its applicability to other continuous t-norms is still to be explored.

References

[1] Franz Baader and Rafael Peñaloza. Are fuzzy description logics with general concept inclusion axioms decidable? In *Proceedings of the 2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pages 1735–1742. IEEE Press, 2011.

[2] Franz Baader and Rafael Peñaloza. GCIs make reasoning in fuzzy DL with the product t-norm undecidable. In Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev, editors, *Proceedings of the 2011 International Workshop on Description Logics (DL'11)*, volume 745 of *CEUR Workshop Proceedings*, 2011.

[3] Franz Baader and Rafael Peñaloza. On the undecidability of fuzzy description logics with GCIs and product t-norm. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Proceedings of 8th International Symposium on Frontiers of Combining Systems (FroCoS 2011)*, volume 6989 of *Lecture Notes in Computer Science*, pages 55–70. Springer-Verlag, 2011.

[4] Fernando Bobillo, Félix Bou, and Umberto Straccia. On the failure of the finite model property in some fuzzy description logics. *CoRR*, abs/1003.1588, 2010.

[5] Fernando Bobillo, Félix Bou, and Umberto Straccia. On the failure of the finite model property in some fuzzy description logics. *Fuzzy Sets and Systems*, 172(1):1–12, 2011.

[6] Fernando Bobillo and Umberto Straccia. A fuzzy description logic with product t-norm. In *Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pages 1–6. IEEE Press, 2007.

[7] Fernando Bobillo and Umberto Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets and Systems*, 160(23):3382–3402, 2009.

[8] Stefan Borgwardt and Rafael Peñaloza. Fuzzy ontologies over lattices with t-norms. In Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev, editors, *Proceedings of the 2011 International Workshop on Description Logics (DL'11)*, volume 745 of *CEUR Workshop Proceedings*, pages 70–80, 2011.

[9] Marco Cerami, Francesc Esteva, and Félix Bou. Decidability of a description logic over infinite-valued product logic. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 203–213. AAAI Press, 2010.

[10] Marco Cerami and Umberto Straccia. On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm. *CoRR*, abs/1107.4212, 2011.