Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 702326, 15 pages
http://dx.doi.org/10.1155/2014/702326

Research Article

Optimised ExpTime Tableaux for $\mathcal{SHIN}$ over Finite Residuated Lattices

Jian Huang, Xinye Zhao, and Jianxing Gong

College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Jian Huang; 13973166586@139.com

Received 25 June 2013; Accepted 16 December 2013; Published 7 April 2014

Academic Editor: Hector Pomares

This study proposes to adopt a novel tableau reasoning algorithm for the description logic $\mathcal{SHIN}$ with semantics based on a finite residuated De Morgan lattice. The syntax, semantics, and logical properties of this logic are given, and a sound, complete, and terminating tableaux algorithm for deciding fuzzy ABox consistency and concept satisfiability problem with respect to TBox is presented. Moreover, based on extended and/or completion-forest with a series of sound optimization technique for checking satisfiability with respect to a TBox in the logic, a new optimized ExpTime (complexity-optimal) tableau decision procedure is presented here. The experimental evaluation indicates that the optimization techniques we considered result in improved efficiency significantly.

1. Introduction

The fuzzy DL (Description Language) over lattice is a generalization of the crisp DL that uses the elements of L as truth values, instead of just the Boolean true and false. Different to fuzzy DLs, elements of the rational unit interval provided a membership degree semantics for their concepts; L is further generalized to address qualitative uncertainty reasoning (by relying, e.g., on {`false`, `likelyfalse`, `unknown`, `likelytrue`, `true`}) and quantitative uncertainty reasoning (by relying, e.g., on $\{0/n, 1/n, \ldots, n/n\}$ (for an integer $n > 1$, in increasing order [1])).

Several attempts have been made at using L-fuzzy set semantics [2], but only a limited kind of semantics over lattices is considered, where conjunction and disjunction are interpreted through the lattice operators *meet* and *join*, respectively. Borgwardt and Peñaloza considered the fuzzy logic $\mathcal{ALCI}$ with semantics based on a finite residuated lattice [3] and a complete De Morgan lattice equipped with a *t-norm* operator [4]. Further, Borgwardt and Peñaloza analysed the consistency and satisfiability problems in the description logic $\mathcal{SHI}$ with semantics based on a complete residuated De Morgan lattice [5] and showed that concept satisfiability in $\mathcal{ALC}$ under this semantics is undecidable, in general, even if a very simple class of infinite lattices is restricted.

In this paper, we extend the more general description logic L-$\mathcal{SHIN}$, where L is a complete De Morgan lattice equipped with a *t-norm* operator. The fuzzy description logic L-$\mathcal{SHIN}$ is a generalization of the crisp description logic $\mathcal{SHIN}$ that uses the elements of L as truth values instead of just the Boolean true and false. We study fuzzy variants of the standard reasoning problems like concept satisfiability and consistency with general concept inclusions (GCI) in this setting and proved that with the help of a tableaux-based algorithm satisfiability becomes decidable and the ExpTime complete if L is required to be finite.

The main contributions of this work can be highlighted twofold as follows.

(i) It presented a novel tableau reasoning algorithm for the description logic $\mathcal{SHIN}$ with semantics based on a finite residuated De Morgan lattice.

(ii) By combining a series of optimization techniques that can be applied to fuzzy DL L-$\mathcal{SHIN}$, our framework reduces the search space of the tableau algorithm more significantly to ExpTime complete.

Table 1: Semantics of L-$\mathcal{SHIN}$ concepts and L-$\mathcal{SHIN}$ roles.

Constructor	Syntax	Semantics
Top	$\top$	$\top^{\mathcal{I}}(x) = 1$
Bottom	$\bot$	$\bot^{\mathcal{I}}(x) = 0$
Conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$
Disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x)$
General negation	$\neg C$	$(\neg C)^{\mathcal{I}}(x) = \sim C^{\mathcal{I}}(x)$
Implication	$C \rightarrow D$	$(C \rightarrow D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)$
Exists restriction	$\exists s \cdot C$	$\exists(s \cdot C)^{\mathcal{I}}(x) = \vee_{y \in \Delta^{\mathcal{I}}} \{s^{\mathcal{I}}(x, y) \Rightarrow C^{\mathcal{I}}(y)\}$
Value restriction	$\forall s \cdot C$	$\forall(s \cdot C)^{\mathcal{I}}(x) = \wedge_{y \in \Delta^{\mathcal{I}}} \{s^{\mathcal{I}}(x, y) \otimes C^{\mathcal{I}}(y)\}$
At-most restriction	$(\leq nR)^{\mathcal{I}}(x)$	$(\leq nR)^{\mathcal{I}}(x) = \wedge_{y_1,\ldots,y_{n+1}} \vee_{i=1}^{n+1} \{\sim R^{\mathcal{I}}(x, y_i)\}$
At-least restriction	$(\geq nR)^{\mathcal{I}}(x)$	$(\geq nR)^{\mathcal{I}}(x) = \vee_{y_1,\ldots,y_n} \wedge_{i=1}^{n} \{R^{\mathcal{I}}(x, y_i)\}$

The paper is organized as follows. Section 2 introduces the syntax and semantics of the fuzzy description logic over lattices L-$\mathcal{SHIN}$ and discusses some logical properties of the logic. In Section 3, we give a detailed presentation of the reasoning algorithm for deciding the consistency and satisfiability of a L-$\mathcal{SHIN}$ ABox, provide the proofs for the termination, soundness, and completeness of the procedure, and address the computational aspect of reasoning in it. In Section 4 we extend the previous results by adding a wide variety of optimisations to achieving a high level of performance on tableau reasoning in expressive description logic L-$\mathcal{SHIN}$. Our empirical evaluation shows that the proposed optimisations result in significant performance improvements in Section 6. Finally, future research issues are outlined in Section 7.

2. The Logic L-$\mathcal{SHIN}$

In this section we introduce a residuated De Morgan lattice extension of the L-$\mathcal{SHIN}$ DL, creating the L-$\mathcal{SHIN}$ language. The fuzzy description logic L-$\mathcal{SHIN}$, which is the extension of $\mathcal{ALC}_L$ [4] and L-$\mathcal{SHI}$ [5], with *the number restrictions* constructor. And L-$\mathcal{SHIN}$ is a generalization of the fuzzy description logic f-$\mathcal{SHIN}$.

Definition 1 (syntax). Let N_C and N_R be pairwise disjoint sets of concepts and role, and let $N_R^+ \subseteq N_R$, N_R be a set of transitive role names. The complex roles are $N_R \cup \{R^- \mid r \in N_R\}$. L-$\mathcal{SHIN}$ complex concepts are defined inductively by the following production rule, where $A \in N_C$ and s is a complex role:

$$C := A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \longrightarrow C_2 \mid \neg C_1 \mid \exists s \cdot C \mid \forall s \cdot C \mid \top \mid \bot . \tag{1}$$

A complex role s is *transitive* if either s or inverse of s belongs to N_R^+.

Definition 2 (semantics). For a residuated De Morgan lattice L, an L interpretation is now a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain for the classical case and $\cdot^{\mathcal{I}}$ is an interpretation function mapping: every concept name C a function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow L$ and every role name r a function $r^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow L$. An *L-assertion* is an expression $\langle a : C \rhd \ell \rangle$, a role assertion of the form $\langle (a, b) : s \rhd \ell \rangle$, where C is a concept, s is a complex role, a, b are individual names, $\ell \in L$, and $\rhd \in \{>, \geq\}$.

A finite set of assertions and terminological axioms is called a *Knowledge Base (KB)*. A fuzzy knowledge base Σ consists of a finite set of axioms organized in three parts: a fuzzy ABox $\mathcal{A}$ (axioms about individuals), a fuzzy TBox $\mathcal{T}$ (axioms about concepts), and a fuzzy RBox $\mathcal{R}$ (axioms about roles); that is, $\Sigma = (\mathcal{A}, \mathcal{R}, \mathcal{T})$. An interpretation $\mathcal{I}$ satisfies (is a model of) a KB Σ if and only if $\mathcal{I}$ satisfies each element in Σ.

An interpretation $\mathcal{I}$ satisfies the concept definition $\langle A \doteq C, \ell \rangle$ if and only if for all $x \in \Delta^{\mathcal{I}}$, $(A^{\mathcal{I}}(x) \Rightarrow C^{\mathcal{I}}(x)) \otimes (C^{\mathcal{I}}(x) \Rightarrow A^{\mathcal{I}}(x)) \geq \ell$. A general TBox is a finite set of GCIs. Σ satisfies the GCI $\langle C \sqsubseteq D, \ell \rangle$ if and only if, for all $x \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq \ell$. An acyclic TBox is a finite set of concept definitions such that every concept name occurs at most once in the lefthand side of an axiom, and there is no cyclic dependency between definitions [6].

The complete set of semantics is depicted in Table 1. Particularly notice that existential and universal quantifiers are not dual to each other, and the implication constructor cannot be expressed in terms of the negation and conjunction.

Next, we will pay more attention to the problem of deciding satisfiability of a concept, especially computing the highest degree with which an individual may belong to a concept.

Definition 3 (ℓ-satisfiable). Let C be L-$\mathcal{SHIN}$ concept descriptions, $\mathcal{T}$ a TBox, and $\ell \in L$. C is ℓ-satisfiable with respect to $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $\vee_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x) \geq \ell$. The best satisfiability degree for C with respect to $\mathcal{T}$ is the largest ℓ such that C is ℓ-satisfiable with respect to $\mathcal{T}$.

The notion of witnessed model for fuzzy DLs was first introduced by Hájek [7]. Witnessed models of fuzzy predicate logic are models in which each quantified formula is

witnessed; that is, the truth value of a universally quantified formula is the minimum of the values of its instances and similarly for existential quantification (maximum) [8].

Definition 4 (witnessed model). Let $n \in \mathbb{N}$. A model I of an KB Σ is n-witnessed if for every $x \in \Delta^{\mathcal{I}}$, every role s, and every concept C there are

$$\begin{aligned}&\left\|(\forall s \cdot C)^{\mathcal{I}}(x)\right\|_M \\ &\quad = \inf_{b \in y_i} \left\|s^{\mathcal{I}}(x, y_i) \otimes C^{\mathcal{I}}(y_i)\right\| = s^{\mathcal{I}}(x,b) \otimes C^{\mathcal{I}}(b), \\ &\left\|(\exists s \cdot C)^{\mathcal{I}}(x)\right\|_M \\ &\quad = \sup_{b \in y_i} \left\|s^{\mathcal{I}}(x, y_i) \Longrightarrow C^{\mathcal{I}}(y_i)\right\| = s^{\mathcal{I}}(x,b) \Longrightarrow C^{\mathcal{I}}(b).\end{aligned} \tag{2}$$

The semantics of the quantifiers require the computation of a supremum or infimum of the membership degrees of a possibly infinite set of elements of the domain. Reasoning is usually restricted to witnessed models in order to obtain effective decision procedures [7]. Since L is finite, we always have the n-witnessed model property for some $n \in \mathbb{N}$. On the basis of $\mathcal{ALC}\mathcal{I}_L$ of [6], we have deduced Lemma 5.

Lemma 5. *If the cardinality of the largest antichain of L is n, then L-$\mathcal{SHIN}$ has the n-witnessed model property.*

For simplification of the algorithm description, we consider $n = 1$. The algorithm and the proofs of correctness can easily be adapted for any other $n \in \mathbb{N}$.

3. Tableau Algorithm for L-$\mathcal{SHIN}$ DL

We assume that all concept descriptions are in *negation normal form (NNF)*; that is, negation appears only in front of concept names. We will also abuse our notation by saying that a TBox $\mathcal{T}$ is in NNF if all the concepts occurring in $\mathcal{T}$ are in NNF. Moreover, we prove the decidability of the L-$\mathcal{SHIN}$ DL by providing a tableaux algorithm for deciding the standard DL inference problems.

3.1. L–$\mathcal{SHIN}$ Tableau. Since most of the inference services of fuzzy DLs can be reduced to the problem of consistency checking for ABoxes [9], here we discuss the approaches to deciding consistency of fuzzy DLs over finite residuated De Morgan lattices in the presence of GCIs. The algorithm we present can be seen as an extension of the tableau presented in [5] and is inspired by the well-known tableau algorithm for f-$\mathcal{SHIN}$ [10], which is the basis for several highly successful implementations.

It is assumed that L is finite, and we can accordingly restrict reasoning to n-witnessed models. In the lattice $L = \langle \zeta, \lhd, \rhd \rangle$, we employ the symbols $\lhd$ and $\rhd$ as a placeholder for the inequalities $\prec, \preceq$ and $\succ, \succeq$ and the symbol $\bowtie$ as a placeholder for all types of inequations. Furthermore we employ the symbols $\lhd^-, \rhd^-$ and $\bowtie^-$ to denote their reflections.

Definition 6. For a fuzzy concept C, we will denote by sub(C) the set that contains C and it is closed under subconcepts of C. The set of all subconcepts of concepts that appear within an ABox is denoted by sub(C). Let R_A be the set of roles occurring in $\mathcal{A}$ and $\mathcal{R}$ together with their inverses, and $I_{\mathcal{A}}$ is the set of individuals in $\mathcal{A}$. A fuzzy tableau $\mathbf{T}$ for Σ is a quadruple in $(\mathbf{S}, \mathcal{G}, \Upsilon, \mathcal{V})$, such that

(i) $\mathbf{S}$ is a nonempty set of individuals (nodes),

(ii) $\mathcal{G}$: $\mathbf{S} \times \text{sub}(A) \to \zeta$ maps each element and concept, that is, a member of sub(A), to the degree of certainty of that element to the concept,

(iii) Υ: $R_\Sigma \times \mathbf{S} \times \mathbf{S} \to \zeta$ maps each role of R_A and pair of elements to the degree of certainty of the pair to the role,

(iv) $\mathcal{V}$: maps individuals occurring in A to elements in S.

For all $s, t \in \mathbf{S}$, $C, D \in \text{sub}(A)$, $\ell \in \zeta$, and $R \in R_A$, $\mathbf{T}$ satisfies the following.

(1) $\Sigma(s, \bot) = 0$, $\Sigma(s, \top) = 1$ for $\forall s \in \mathbf{S}$.

(2) If $\langle \neg C, \bowtie, \ell \rangle \in \mathcal{G}(s)$, then $\langle C, \bowtie^-, \sim \ell \rangle \in \mathcal{G}(s)$.

(3) If $\langle C \sqcap D, \rhd, \ell \rangle \in \mathcal{G}(s)$, then $\langle C, \rhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle D, \rhd, \ell \rangle \in \mathcal{G}(s)$.

(4) If $\langle C \sqcup D, \lhd, \ell \rangle \in \mathcal{G}(s)$, then $\langle C, \lhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle D, \lhd, \ell \rangle \in \mathcal{G}(s)$.

(5) If $\langle C \sqcup D, \rhd, \ell \rangle \in \mathcal{G}(s)$, then $\langle C, \rhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle D, \rhd, \ell \rangle \in \mathcal{G}(s)$.

(6) If $\langle C \sqcap D, \lhd, \ell \rangle \in \mathcal{G}(s)$, then $\langle C, \lhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle D, \lhd, \ell \rangle \in \mathcal{G}(s)$.

(7) If $\langle \forall R \cdot C, \rhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle \langle s, t \rangle, \rhd', \ell_1 \rangle \in \Upsilon(R)$ is conjugated with $\langle \langle s, t \rangle, \rhd^-, \sim \ell \rangle$, then $\langle \forall C, \rhd, \ell \rangle \in \mathcal{G}(t)$.

(8) If $\langle \forall R \cdot C, \lhd, \ell \rangle \in \mathcal{G}(s)$ and $\langle \langle s, t \rangle, \rhd, \ell_1 \rangle \in \Upsilon(R)$ is conjugated with $\langle \langle s, t \rangle, \lhd, \sim \ell \rangle$, then $\langle \forall C, \lhd, \ell \rangle \in \mathcal{G}(t)$.

(9) If $\langle \exists R \cdot C, \rhd, \ell \rangle \in \mathcal{G}(s)$, then there exists $t \in \mathbf{S}$ such that $\langle \langle s, t \rangle, \rhd, \ell \rangle \in \Upsilon(R)$ and $\langle \langle s, t \rangle, \rhd, \sim \ell \rangle$, then $\langle \forall C, \rhd, \ell \rangle \in \mathcal{G}(t)$.

(10) If $\langle \forall R \cdot C, \lhd, \ell \rangle \in \mathcal{G}(s)$, then there exists $t \in \mathbf{S}$ such that $\langle \langle s, t \rangle, \lhd^-, \sim \ell \rangle \in \Upsilon(R)$ and $\langle \langle s, t \rangle, \lhd, \sim \ell \rangle$, then $\langle \forall C, \rhd, \ell \rangle \in \mathcal{G}(t)$.

(11) If $\langle \exists s \cdot C, \lhd, \ell \rangle \in \mathcal{G}(s)$, and $\langle \langle s, t \rangle, \rhd, \ell_1 \rangle \in \Upsilon(R)$ is conjugated with $\langle s \cdot C, \lhd, \ell \rangle$, for some $R \sqsubseteq^*$ with $\texttt{Trans}(R)$, then $\langle \exists s \cdot C, \lhd, \ell \rangle \in \mathcal{G}(s)$.

(12) If $\langle \forall s \cdot C, \rhd, \ell \rangle \in \mathcal{G}(s)$, and $\langle \langle s, t \rangle, \rhd', \ell_1 \rangle \in \Upsilon(R)$ is conjugated with $\langle s \cdot C, \rhd^-, \sim \ell \rangle$, for some $R \sqsubseteq^*$ with $\texttt{Trans}(R)$, then $\langle \forall s \cdot C, \rhd, \ell \rangle \in \mathcal{G}(s)$.

(13) $\langle \langle s, t \rangle, \bowtie, \ell \rangle \in \Upsilon(R)$ if $\langle \langle t, s \rangle, \bowtie, \ell \rangle \in \Upsilon(R)(\texttt{Inv}(R))$.

(14) If $\langle \langle s, t \rangle, \rhd, \ell \rangle \in \Upsilon(R)$ and $R \sqsubseteq^* S$ then, $\langle \langle s, t \rangle, \rhd, \ell \rangle \in \Upsilon(S)$.

(15) $\langle \geq pR, \rhd, \ell \rangle \in \Upsilon(R)$, then $\{t \in \mathbf{S} \mid \langle \langle s, t \rangle, \rhd, \ell \rangle \in \Upsilon(R)\} \geq p$.

(16) $\langle \leq pR, \triangleleft, \ell\rangle \in \Upsilon(R)$, then $\{t \in \mathbf{S} \mid \langle\langle s,t\rangle, \triangleleft^{-}, \sim \ell\rangle \in \Upsilon(R)\} \geq p+1$.

(17) $\langle \geq pR, \triangleleft, \ell\rangle \in \Upsilon(R)$, then $\{t \in \mathbf{S} \mid \langle\langle s,t\rangle, \triangleright, \ell_i\rangle \in \Upsilon(R)\} \leq p-1$, conjugated with $\langle\langle s,t\rangle, \triangleleft, \ell\rangle$.

(18) $\langle \leq pR, \triangleright, \ell\rangle \in \Upsilon(R)$, then $\{t \in \mathbf{S} \mid \langle\langle s,t\rangle, \triangleright', \ell_i\rangle \in \Upsilon(R)\} \leq p$, conjugated with $\langle\langle s,t\rangle, \triangleright^{-}, \sim \ell\rangle$.

(19) There do not exist two conjugated triples in any label of any individual $x \in \mathbf{S}$.

(20) If $\langle a : C \bowtie \ell\rangle \in A$, then $\langle C \bowtie \ell\rangle \in \mathcal{G}(\mathcal{V}(a))$.

(21) If $\langle (a,b) : R \bowtie \ell\rangle \in A$, then $\langle (\mathcal{V}(a), (\mathcal{V}(b)) \bowtie \ell\rangle \in \Upsilon(R)$.

(22) If $a \neq b \in \mathcal{A}$, then $\mathcal{V}(a) \neq \mathcal{V}(b)$.

Lemma 7. *An L-$\mathcal{SHIN}$ ABox $\mathcal{A}$ is consistent with respect to $\mathcal{R}$ if and only if there exists a fuzzy tableau for $\mathcal{A}$ with respect to $\mathcal{R}$.*

Proof. Similar to that of Lemma 6.5 of [11], here gives the proof of the Lemma 7. For the direction if $\mathbf{T} = (\mathbf{S}, \mathcal{G}, \Upsilon, \mathcal{V})$ is a fuzzy tableau for an ABox $\mathcal{A}$ with respect to $\mathcal{R}$, we can construct a fuzzy interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that is a model of $\mathcal{A}$. Consider the following:

$$\begin{aligned}
&\Delta^{\mathcal{I}} = \mathbf{S},\\
&a^{\mathcal{I}} = \mathbf{V}(a), \quad a \in \mathcal{I}_A,\\
&\top^{\mathcal{I}} = \mathcal{G}(s,\top), \quad \forall s \in \mathbf{S},\\
&\bot^{\mathcal{I}} = \mathcal{G}(s,\bot), \quad \forall s \in \mathbf{S},\\
&A^{\mathcal{I}} = \mathcal{G}(s,A), \quad \forall s \in \mathbf{S} \text{ and concept names } A.
\end{aligned} \tag{3}$$

For all role $R \in \mathbf{R}_A$,

$$R^{\mathcal{I}}(s,t) = \begin{cases} R^{+}_{\Upsilon}(s,t), \forall \langle s,t\rangle \in \mathbf{S}\times\mathbf{S} & \text{if } \mathtt{Trans}(R)\\ R_{\Upsilon}(s,t), \forall \langle s,t\rangle \in \mathbf{S}\times\mathbf{S} & \text{otherwise,}\end{cases} \tag{4}$$

where $R_{\Upsilon}(s,t)$ is a binary fuzzy relation defined as $R_{\Upsilon}(s,t) = \Upsilon(R, \langle s,t\rangle)$ for all $\forall\langle s,t\rangle \in \mathbf{S}\times\mathbf{S}$, and $R^{+}_{\Upsilon}(s,t)$ represents its sup-min transitive closure [12]. The proof of this property is quite technical and omitted here. From all above properties, the interpretation of individuals and roles implies that $\mathcal{I}$ satisfies each assertion in $\mathcal{A}$ inductively. □

3.2. Constructing a L-$\mathcal{SHIN}$ Tableau. Like most of the tableaux algorithms, our algorithm works on *completion forests* rather than on completion trees which is applied in work [5]. As pointed out in [13], that is because an ABox might contain several individuals with arbitrary roles connecting them.

Definition 8 (completion forest). A completion forest $\mathcal{F}$ for Σ is a collection of trees whose distinguished roots are arbitrarily connected by edges. Each node x is labelled with a set $\Sigma(x) = \{\langle C, \bowtie, l\rangle\}$. Each edge $\langle x,y\rangle$ is labelled with a set $\Sigma(\langle x,y\rangle) = \{\langle R, \bowtie, l\rangle\}$, where $R \in R_A$ are roles and its inverse occurring in $\mathcal{A}$. A completion forest comes with an explicit inequality relation $\neq$ on nodes and an explicit equality relation $\doteq$ which are implicitly assumed to be symmetric. Given a completion forest, If nodes x and y are connected by an edge $\langle v,w\rangle$ with $\langle R,l\rangle$ occurring in $\Sigma(\langle x,y\rangle)$, then y is called an R-successor of x and x is called an R-predecessor of y. A node x is a positive (resp., negative) successor (resp., predecessor or neighbour) of y.

Table 2: Conjugated pairs.

	$\langle \alpha \not\succeq l'\rangle$	$\langle \alpha \prec l'\rangle$
$\langle \alpha \succ l\rangle$	$\neg(\exists l'' \cdot l'' \succ l \wedge l'' \not\succeq l')$	$l \not\succ l'$
$\langle \alpha \not\preceq l\rangle$	$\neg(\exists l'' \cdot l'' \not\preceq l \wedge l'' \not\succeq l')$	$\neg(\exists l'' \cdot l'' \not\preceq l \wedge l'' \prec l')$

A node is blocked if it is either directly or indirectly blocked. It is indirectly blocked if its *R-predecessor* is either directly or indirectly blocked. A node x is directly blocked if and only if none of its ancestors are blocked, and it has an ancestor y such that $\Sigma(x) = \Sigma(y)$.

A node x is said to contain a clash satisfying one of the conditions such that if and only if there exist two conjugated triples in $\Sigma(\langle x,y\rangle)$, or if $\Sigma(\langle x,y\rangle) \cup \{\langle \mathtt{Inv}(R), \bowtie, \ell\rangle \mid \langle R, \bowtie, \ell\rangle \in \Sigma(\langle y,x\rangle)\}$ and x, y are root nodes, contains two conjugated triples. Table 2 shows all the entries under which condition the row column pair of L-constraints is a *conjugated pair* (e.g., $a : A \succ u$ and $a : A \not\succeq v$ is a conjugated pair as $\neg(\exists v \cdot v \succ u \wedge v \not\succeq v)$).

The calculus is based on a set of constraint propagation rules transforming a set S into satisfiability preserving sets S_i until either one of S_i contains an inconsistency (clash), or some S_i is completed and clash-free; that is, no rule can further be applied to S_i and S_i contains no clash (indicating that from S_i a model of S can be built).

$\mathcal{F}$ is then expanded by repeatedly applying the rules from Table 3. The notation R^{*} denotes either the role R or the role returned by $\mathtt{Inv}(R)$, and the notation $\langle *, \bowtie, \ell\rangle$ denotes any role that participates in such a triple.

The completion forest is complete when for some node $x : \Sigma(x)$ contains a clash, or none of the completion rules in Table 3 are applicable. The algorithm is correct in the sense that it produces a clash if and only if Σ is inconsistent. The expansion rules are based on the properties of the semantics presented in Definition 6.

3.3. Soundness and Completeness of the L-$\mathcal{SHIN}$ Tableaux Algorithm

Lemma 9 (termination, soundness, completeness). *For a given KB Σ:*

(i) *the tableau algorithm terminates;*

(ii) *if the expansion rules can be applied to Σ such that they yield a complete completion-forest $\mathcal{F}$ such that $C_{\mathcal{F}}$ has a solution, then Σ has a model;*

(iii) *if Σ has a model, then the expansion rules can be applied in such a way that the tableaux algorithm yields a complete completion forest for Σ such that $C_{\mathcal{F}}$ has a solution.*

Table 3: Rule description.

Rule	Description
$(\neg_{\bowtie})$	**If** $\langle \neg C, \bowtie, l\rangle \in \Sigma(x)$ **and** $\langle C, \bowtie^{-}, \sim \ell\rangle \notin \Sigma(x)$, **then** $\Sigma(x) \rightarrow \Sigma(x) \cup \langle C, \bowtie^{-}, \sim \ell\rangle$
$(\sqcap_{\triangleright})$	**If** $\langle C_1 \sqcap C_2, \triangleright, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked **and** $\{\langle C_1, \triangleright, \ell\rangle, \langle C_2, \triangleright, \ell\rangle\} \notin \Sigma(x)$, **then** $\Sigma(x) \rightarrow \Sigma(x) \cup \{\langle C_1, \triangleright, \ell\rangle, \langle C_2, \triangleright, \ell\rangle\}$
$(\sqcup_{\triangleleft})$	**If** $\langle C_1 \sqcup C_2, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked **and** $\{\langle C_1, \triangleleft, \ell\rangle, \langle C_2, \triangleleft, \ell\rangle\} \notin \Sigma(x)$, **then** $\Sigma(x) \rightarrow \Sigma(x) \cup \{\langle C_1, \triangleleft, \ell\rangle, \langle C_2, \triangleleft, \ell\rangle\}$
$(\sqcup_{\triangleright})$	**If** $\langle C_1 \sqcup C_2, \triangleright, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked **and** $\{\langle C_1, \triangleright, l\rangle, \langle C_2, \triangleright, \ell\rangle\} \cap \Sigma(x) = \emptyset$, **then** $\Sigma(x) \rightarrow \Sigma(x) \cup \{C\}$ for some $C \in \{\langle C_1, \triangleright, \ell\rangle, \langle C_2, \triangleright, \ell\rangle\}$
$(\sqcap_{\triangleleft})$	**If** $\langle C_1 \sqcap C_2, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked **and** $\{\langle C_1, \triangleleft, \ell\rangle, \langle C_2, \triangleleft, \ell\rangle\} \cap \Sigma(x) = \emptyset$, **then** $\Sigma(x) \rightarrow \Sigma(x) \cup \{C\}$ for some $C \in \{\langle C_1, \triangleleft, \ell\rangle, \langle C_2, \triangleleft, \ell\rangle\}$
$(\exists_{\triangleright})$	**If** $\langle \exists R \cdot C, \triangleright, \ell\rangle \in \Sigma(x)$, x is not blocked **and** x has no R-neighbour y connected with a triple $\langle P^{*}, \triangleright, \ell\rangle$, $P \sqsubseteq^{*} R$ and $\langle C, \triangleright, \ell\rangle \in \Sigma(y)$ **, then** create a new node y with $\Sigma(\langle x, y\rangle) = \{\langle R, \triangleright, \ell\rangle\}$, $\Sigma(y) = \{\langle C, \triangleright, \ell\rangle\}$
$(\forall_{\triangleleft})$	**If** $\langle \forall R \cdot C, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not blocked **and** x has no R-neighbour y connected with a triple $\langle P^{*}, \triangleleft^{-}, \sim \ell\rangle$, $P \sqsubseteq^{*} R$ and $\langle C, \triangleleft, \ell\rangle \in \Sigma(y)$**, then** create a new node y with $\Sigma(\langle x, y\rangle) = \{\langle R, \triangleleft^{-}, \sim \ell\rangle\}$, $\Sigma(y) = \{\langle C, \triangleleft, \ell\rangle\}$
$(\forall_{\triangleright})$	**If** $\langle \forall R \cdot C, \triangleright, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked, **and** x has an R-neighbour y with $\langle C, \triangleright, \ell\rangle \notin \Sigma(y)$ **and** $\langle *, \triangleright^{-}, \sim \ell\rangle$ is conjugated with the positive triple that connects x and y, **then** $\Sigma(y) \rightarrow \Sigma(y) \cup \langle C, \triangleright, \ell\rangle$
$(\exists_{\triangleleft})$	**If** $\langle \exists R \cdot C, \triangleleft, l\rangle \in \Sigma(x)$, x is not indirectly blocked **and** x has an R-neighbour y with $\langle C, \triangleright, \ell\rangle \notin \Sigma(y)$ **and** $\langle *, \triangleleft, \ell\rangle$ is conjugated with the positive triple that connects x and y, **then** $\Sigma(y) \rightarrow \Sigma(y) \cup \langle C, \triangleleft, \ell\rangle$
$(\forall_{+})$	**If** $\langle \forall R \cdot C, \rightarrow, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked, x has a P-neighbour y with, $\langle C, \triangleright, \ell\rangle \notin \Sigma(y)$, **and** there is some P, with $\mathtt{Trans}(P)$, and $P \sqsubseteq^{*} R$, $\langle \forall P \cdot C, \triangleright, \ell\rangle \notin \Sigma(y)$ **and** $\langle *, \triangleright^{-}, \sim \ell\rangle$ is conjugated with the positive triple that connects x and y, **then** $\Sigma(y) \rightarrow \Sigma(y) \cup \langle \forall P \cdot C, \triangleright, l\rangle$
$(\exists_{+})$	**If** $\langle \exists R \cdot C, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked, x has a P-neighbour y with, $\langle C, \triangleright, \ell\rangle \notin \Sigma(y)$, **and** there is some P, with $\mathtt{Trans}(P)$, and $P \sqsubseteq^{*} R$, $\langle \exists P \cdot C, \triangleleft, \ell\rangle \notin \Sigma(y)$ **and** $\langle *, \triangleleft, \sim \ell\rangle$ is conjugated with the positive triple that connects x and y, **then** $\Sigma(y) \rightarrow \Sigma(y) \cup \langle \exists P \cdot C, \triangleleft, \ell\rangle$
$(\geq_{\triangleright})$	**If** $\langle \geq p \cdot R, \triangleright, l\rangle \in \Sigma(x)$, x is not blocked, **and** there are no p R-neighbours $y_1, \ldots, \ell_p$, connected to x with a triple $\langle \exists P^{*}, \triangleright, l\rangle \notin \Sigma(y)$, $P \sqsubseteq^{*} R$, **and** $y_i \neq y_j$ for $1 \leq i \leq j \leq p$, **then** create p new nodes $y_1, \ldots, \ell_p$, with $\Sigma(\langle x, y_i\rangle) = \{\langle R, \triangleright, \ell\rangle\}$ and $y_i \neq y_j$ for $1 \leq i \leq j \leq p$
$(\leq_{\triangleleft})$	**If** $\langle \leq p \cdot R, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not blocked, **then** apply $(\exists_{+})$ rule for the triple $\langle \geq (p+1)R, \triangleleft^{-}, \sim \ell\rangle$
$(\leq_{\triangleleft})$	**If** $\langle \leq p \cdot R, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not blocked, **then** apply $(\exists_{+})$ rule for the triple $\langle \geq (p+1)R, \triangleleft^{-}, \sim \ell\rangle$
$(\leq_{\triangleright})$	**If** $\langle \leq p \cdot R, \triangleright, l\rangle \in \Sigma(x)$, x is not indirectly blocked, **and** there are $p+1$ R-neighbours $y_1, \ldots, y_{p+1}$ connected to x with a triple $\langle P^{*}, \triangleright', \sim \ell_i\rangle$, $P \sqsubseteq^{*} R$, **and** which is conjugated with $\langle P^{*}, \triangleright^{-}, \sim \ell\rangle$, and there are two of them y, z, with no $y \neq z$, **and** y is neither a root node nor an ancestor of z, **then** $\Sigma(z) \rightarrow \Sigma(z) \cup \Sigma(y)$, **and** if z is an ancestor of x, then $\Sigma(\langle z, x\rangle) \rightarrow \Sigma(\langle z, x\rangle) \cup \mathtt{Inv}(\Sigma(\langle x, y\rangle))$ else $\Sigma(\langle x, z\rangle) \rightarrow \Sigma(\langle x, z\rangle) \cup \Sigma(\langle x, y\rangle)$, **and** $\Sigma(\langle x, y\rangle) \rightarrow \emptyset$, **and** Set $u \neq z$ for $\forall u$ with $u \neq y$
$(\geq_{\triangleleft})$	**If** $\langle \geq p \cdot R, \triangleleft, \ell\rangle \in \Sigma(x)$, x is not indirectly blocked, **then** apply $(\leq_{\triangleright})$ rule for the triple $\langle \leq (p-1)R, \triangleleft^{-}, \sim \ell\rangle$
$(\leq_{r\triangleright})$	**If** $\langle \leq p \cdot R, \triangleright, \ell\rangle \in \Sigma(x)$, **and** there are $p+1$ R-neighbours $y_1, \ldots, y_{p+1}$ connected to x with a triple $\langle P^{*}, \triangleright', \sim \ell_i\rangle$, $P \sqsubseteq^{*} R$, **and** conjugated with $\langle P^{*}, \triangleright^{-}, \sim \ell\rangle$, and there are two of them y, z, both root nodes, with no $y \neq z$, **then** $\Sigma(z) \rightarrow \Sigma(z) \cup \Sigma(y)$, **and** for all edges $\langle y, w\rangle$: (1) if the edge $\langle z, w\rangle$ does not exist, create it with $\Sigma(\langle z, w\rangle) \rightarrow \emptyset$; (2) $\Sigma(\langle z, w\rangle) \rightarrow \Sigma(\langle z, w\rangle) \cup \Sigma(\langle y, w\rangle)$, **and** for all edges $\langle w, y\rangle$: (1) if the edge $\langle w, z\rangle$ does not exist, create it with $\Sigma(\langle w, z\rangle) \rightarrow \emptyset$; (2) $\Sigma(\langle w, z\rangle) \rightarrow \Sigma(\langle w, z\rangle) \cup \Sigma(\langle w, y\rangle)$ **and** Set $\Sigma(y) = \emptyset$, and remove all edges to/from y, **and** set $u \neq z$ for $\forall u$ with $u \neq y$ and set $y = z$
$(\geq_{r\triangleleft})$	$\langle \geq p \cdot R, \triangleleft, \ell\rangle \in \Sigma(x)$, **then** apply $(\leq_{r\triangleright})$ rule for the triple $\langle \leq (p-1)R, \triangleleft^{-}, \sim \ell\rangle$

3.4. Computational Complexity of the Tableau Algorithm for L-$\mathcal{SHIN}$. We have described a tableau procedure for reasoning in L-$\mathcal{SHIN}$. Due to the tableau rules are nondeterministic, every application of expansion rules to a termination after at most exponentially many rule applications. So according to the theorem in [5] that the tableaux algorithm is NExpTime complete, we get the following proposition deciding the computational complexity of the tableau algorithm for L-$\mathcal{SHIN}$.

Proposition 10. *The concept satisfiability checking problem in L-$\mathcal{SHIN}$ with respect to witnessed models can be decided in NExpTime.*

Proof (sketch). It is assumed that L-$\mathcal{SHIN}$ has the n-witnessed model property for some $n \geq 1$, the completion forest has to generate n different successors for every existential ($\exists$) and universal ($\forall$) restriction to ensure that the degrees guessed for these complex concepts are

indeed witnessed by the model. As stated by Table 1, we have to introduce n individuals $\{y_1, \ldots, y_n\}$ and $2n$ values $\{\ell_1^1, \ldots, \ell_1^n\} \in L, \{\ell_2^1 \ldots, \ell_2^n\} \in L$, which satisfies $\bigvee_{i=1}^{n} \ell_1^i \otimes \ell_2^i = \ell$ or $\bigwedge_{i=1}^{n} \ell_1^i \Rightarrow \ell_2^i = \ell$, respectively. In the following we let $n_{\max}$ be the number that occurs in a number restriction, and let h be the number of different lattices appearing in $\mathscr{A}$. □

Following [14] we set $n = ||A| + |R|| = \mathcal{O}(|A| + |R|) = \mathcal{O}(2n)$; in the meantime, we let $m = |\mathrm{sub}(D)| = \mathcal{O}(2|A||R|) = \mathcal{O}(n^2)$, $k = |R_A| = \mathcal{O}(|A| + |R|) = \mathcal{O}(2n)$, $n_{\max} = \mathcal{O}(2|A|) = \mathcal{O}(2n)$, $h = \mathcal{O}(|A|) = \mathcal{O}(n)$, and $q = \max\{D_{\mathscr{L}}(s) \mid \forall s \in \mathscr{T}\}$. Then due to the proof of Lemma 6.9 in [11] and by the addition of the number of different lattices appearing in $\mathscr{A}$-q: a completion forest for $\mathscr{A}$ becomes no longer than $2^{8 \cdot m \cdot h \cdot k \cdot q}$ and that the outdegree is bounded by $2 \cdot h \cdot m \cdot n_{\max}$. Consequently, the L-$\mathcal{SHIN}$ algorithm will construct a completion forest with no more than $(2 \cdot h \cdot m \cdot n_{\max})^{2^{8 \cdot m \cdot h \cdot k \cdot q}} = \mathcal{O}((2 \cdot n \cdot n^2 \cdot 2^n)^{2^{8 \cdot n^2 \cdot n \cdot n \cdot q}}) = \mathcal{O}(2^{n \cdot 2^{8 \cdot n^4 \cdot q}}) = \mathcal{O}(2^{2^{q \cdot n^4}})$ nodes.

For fuzzy description logics with semantics based on complete residuated De Morgan lattices (e.g., L-$\mathcal{ALC}$), strong satisfiability with respect to general TBoxes is undecidable for some infinite lattices while, for finite lattices, decidability is regained [5]. If one considers the $\mathcal{ALC}$ without terminological axioms, concept satisfiability is ExpTime complete as in the crisp case [15]. The problem with respect to general TBoxes becomes ExpTime complete matching the complexity of the crisp case, though arbitrary (finite) lattices and t-norms are allowed [4]. Besides, strong concept satisfiability is in NExpTime L-$\mathcal{SHI}$.

When we consider finite De Morgan lattices L, then satisfiability problem can be effectively decided. We guess that the computational complexity of it can be improved to ExpTime either by an ABox partitioning [16] or with the help of global caching and other techniques [17].

4. Optimization Techniques Employed in Tableau Algorithm for L-$\mathcal{SHIN}$

From Proposition 10, it is obvious that the theoretical complexity of the tableau reasoning algorithm for L-$\mathcal{SHIN}$ is 2-NExptime, which is too expensive for a reasoner to deal with. So we plan to investigate some optimizations [18] developed for tableaux algorithms for crisp DLs, which can be transferred to our setting to reduce the search space created by the choice of lattice values. By using these techniques, a theoretically expensive computation could be converted to an equivalent of practically lower complexity.

In this section we describe in detail the optimisation techniques employed in tableau algorithm for L-$\mathcal{SHIN}$. Not only focusing mainly on novel techniques and significant refinements and extensions of previously known techniques such as global caching, we also apply some simple but often very effective optimisations to our implement. First of all, we perform some preprocessing optimisations directly on the syntax of the input. Then in virtue of the ideas of both backjumping and global caching, we investigate an optimized tableau-based reasoning algorithm and prove its ExpTime complete.

4.1. Preprocessing Optimisations. In case of obvious clash detection or cycle via concept names, preprocessing optimisations (e.g., *absorption* or *told cycle elimination*) can lead to important speedup of the subsequent reasoning process. These optimisations serve to preprocess and simplify the input into a form more amenable to later processing [18]. Furthermore, these optimisations are not specific to tableau-based fuzzy reasoners and may also be useful with other kinds of fuzzy DL reasoners. Herein on the basis of L-$\mathcal{SHIN}$, we apply two striking optimizations techniques to our design and implement.

4.1.1. Partition Based on Connectivity. Partition based on connectivity is a quite effective optimization technique to boost up the performance of reasoning, which can be applied to any fuzzy DL without nominals independently of the fuzzy operators used to provide the interpretations [17]. Termination of the expansion of the completion forest is a result of the properties of the expansion rules in Table 3, and a tableau expansion rule can either add (i) a new neighbour node to the node of examination, (ii) new membership triples in this node, or (iii) new membership triples to neighbouring nodes. The expansion rule will examine the different ways that the tableau expansion rules affect this completion forest, with the respect of different connectivity circumstances. Inspired by the work in [17], we have the following definitions.

Definition 11. Let $a, b, c \in \mathbf{I}$, $\ell \in L$. The connection relation between two individuals a, b in an ABox $\mathscr{A}$ is inductively defined through role R_A:

$$a \rightsquigarrow_{\mathscr{A}} b \Longleftrightarrow \begin{cases} R_A(a,b) \bowtie l \in \mathscr{A} & \text{derectly connected} \\ (a \rightsquigarrow_{\mathscr{A}} c) \cup (b \rightsquigarrow_{\mathscr{A}} c) & \text{inderectly connected.} \end{cases} \tag{5}$$

Note that c may be run into an iterative process for connecting indirectly. The individuals in the ABox $\mathscr{A}$ may be divided into one or more individual groups. Then individual group is partitioned into one or more independent subsets called *Assertion Groups (AG)* [16]. For instance, two assertions A_1 and A_2 are in the same AG if A_1 is directly or indirectly inferred from A_2 (through the application of completion rules), or A_1 and A_2 differ only in terms of their certainty values and/or conjunction and disjunction functions. Each group is composed of individuals that are connected to each other through role assertions R_A.

Definition 12. We denote $AG_{[a]}$ with the partition of the ABox $\mathscr{A}$ that contains only connected individuals in $\mathscr{A}$

$$AG_{[a]} = \{a\} \cup \{x \mid \forall x \in \mathbf{I}, a \rightsquigarrow_{\mathscr{A}} x\}. \tag{6}$$

Proposition 13. *It holds that*

(1) $\cup AG_i = \mathscr{A}$,

(2) $AG_i \cap AG_j = \emptyset$, *for each pair* $AG_i, AG_j \in A$ *such that* $AG_i = AG_j$,

(3) $\mathscr{A}$ *is consistent with respect to a TBox* $\mathscr{T}$ *if and only if each* AG_i *is consistent with respect to a TBox* $\mathscr{T}$.

Partition based on connectivity optimization technique has several advantages. First, it is of polynomial complexity and it can be applied to any fuzzy DL. In addition, If any AG is found to be inconsistent, this implies that the whole $\mathscr{A}$ is inconsistent. A related benefit is to more precisely identify the assertions that cause the inconsistency. Finally, the speed of solving a few small constraint sets would be faster than solving one large constraint set.

4.1.2. Dealt with TBox Axioms. If dealt with axioms of TBox $\mathscr{T}$ naively, which maybe lead to a serious degradation in reasoning performance, as each such axiom would cause a disjunction to be added to every node of the completion forest, leading to potentially enormous amounts on nondeterministic expansion [18].

Given $\varsigma \in \mathscr{T}$, with $\varsigma = ((C_1 \vee D_1) \wedge \cdots \wedge (C_n \vee D_n))$, testing the satisfiability of ς leads to the construction of a completion forest containing k nodes, then there are 2^{kn} different ways to apply the $\sqcap$- and $\sqcup$-rules to the resulting k copies of A, which leads to the explosion in the size of the search space. Fortunately, some notable works of optimisations known as *lazy unfolding* and *absorption* had been used to identifying these problems [19]. Herein we choose *lazy unfolding* optimisation to address this issue. Due to the limitation of the paper, detailed definition of *lazy unfolding* is omitted here, and we refer the reader to the above literatures.

4.2. Core Satisfiability Optimisations. In the following, we present core reasoning of L-$\mathcal{SHIN}$, a satisfiability checking algorithm, which implements a highly optimised version on the basis of the tableau algorithm described in Section 3. Before introducing the optimisations extended to the naive tableau algorithm, we present a brief introduction to *backjumping* and *global caching* and show how such algorithms are implemented in practice.

4.2.1. Dependency Directed Backtracking (Backjumping). Backjumping is an optimisation that is crucial for effective tableau based reasoning. Inherent unsatisfiability concealed in subformulae can lead to large amounts of unproductive backtracking search known as thrashing, especially if the unsatisfiability caused by the modal subformulae had been slightly less trivial [20]. Consider the following modified formula ς:

$$\varsigma = (p_1 \vee q_1) \wedge \cdots \wedge (p_n \vee q_n) \wedge (\neg \exists R \cdot (A, B)) \wedge (\forall R \cdot (\neg A)). \quad (7)$$

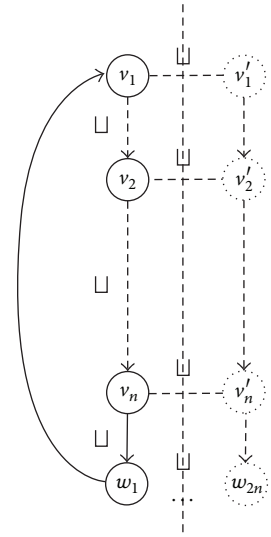

Figure 1: Runing the search using backjumping.

To avoid an exponential (2^n) search in the case of ς, a form of *dependency directed backtracking* called *backjumping* can be adapted, which has also been used, for example, in solving constraint satisfiability problems [21] and (in a slightly different form) in the HARP theorem prover [22]. The arithmetic labels each formula ς in a node v with a dependency set $\mathrm{dep}(\varsigma, v)$ indicating the branching points on which it depends.

In case node$v$ is with $\mathrm{dep}(\varsigma, \neg\varsigma) \in \Sigma(v)$, $\mathrm{dep}(\varsigma, v)$ and $\mathrm{dep}(\neg\varsigma, v)$ are used to identify the most recent branching point b on which ς or $\neg\varsigma$ depends. If the dependance exists, the algorithm can then jump back to b over intervening branching points without exploring any alternative branches (nondeterministic choices) and make a different nondeterministic choice which might not lead to the same closure condition being encountered. Figure 1 illustrates how the expansion below v is pruned by backjumping, with the number of *R-successors* explored being reduced by $2^n - 1$. We will go into particulars in Algorithm 2. In case no such b exists, the clash did not depend on any nondeterministic choice, and the algorithm stops, reporting the unsatisfiability of the TBox [20].

4.2.2. Global Caching. In order to avoid the same subproblem being solved again and again, it is possible to cache and reuse the results of such subproblems. The usual technique is to use a hash table to store the satisfiability status of node labels (i.e., sets of concepts treated as a conjunction). At present, there are various caching methods briefly outlined in [23]:

(i) no caching DFS (NC),

(ii) unsat caching DFS (UC),

(iii) mixed caching DFS (MC),

(iv) unrestricted global caching and propagation (GC-NonDFS),

(v) global caching DFS with propagation (GC-DFS).

Data: Let $t \in \mathbb{N}^+, t \approx m/2, \overline{d}, d_t \in N^\Sigma, \overline{d} = d_t$.
Result: $\text{glb}(\Sigma, \alpha) = \overline{d}$.
begin
  while $\|N^\Sigma\| > 1$ **do**
    if $\Sigma \models \langle \alpha \geq d_t \rangle$ **then**
      $\overline{d} = d_t$
      $N^\Sigma \leftarrow N^\Sigma \setminus \{d' : d' \ngeq \overline{d} \cup \{\overline{d}\}\}$
      $d_t \leftarrow d_{t+1}$
    else
      $N^\Sigma \leftarrow N^\Sigma \setminus \{d' : d' \geq \overline{d}\}$
      $d_t \leftarrow d_{t-1}$

Algorithm 1: Calculate glb.

Global caching means that each possible set of concepts/formulas is expanded at most once. With respect to the size of the given problem, the notion of global caching is an ExpTime complexity procedure if the search space contains at most an exponential number of different nodes and as long as the extension of each node requires at most exponential time.

4.2.3. Optimized GLB. From Table 2, we can conclude that an L-$\mathcal{SHIN}$ ABox $\mathcal{A}$ can contain a number of positive or negative assertions without forming a contradiction. Therefore, it is useful to compute lower and upper bounds of finite lattice degree.

Given a fuzzy KB Σ and an assertion α, it is absorbing to compute best lower and upper truth-value bounds of α. To this end we define the *greatest lower bound* of α with respect to Σ (denoted by $\text{glb}(\Sigma, \alpha)$) to be $\oplus\{l \mid \Sigma \models \langle \alpha \geq l \rangle\}$. Similarly, we define the *least upper bound* of α with respect to Σ (denoted by $\text{lub}(\Sigma, \alpha)$) to be $\otimes\{l \mid \Sigma \models \langle \alpha \leq l \rangle\}$. $\text{lub}(\Sigma, \alpha))$ could be computed from the glb as $\text{lub}(\Sigma, \alpha)) = \sim \text{glb}(\Sigma, \sim \alpha))$ because $\sim \text{lub}(\Sigma, \alpha)) = \sim \otimes\{l \mid \Sigma \models \langle \alpha \leq l \rangle\} = \otimes\{\sim l \mid \Sigma \models \langle \alpha \leq l \rangle\} = \otimes\{l \mid \Sigma \models \langle \alpha \leq \sim l \rangle\} = \oplus\{l \mid \Sigma \models \langle \sim \alpha \geq l \rangle\} = \sim \text{glb}(\Sigma, \sim \alpha))$. Determining the lub and the glb is called the *Best Lattice Degree Bound* (BLDB) problem.

Concerning the BCVB problem, we may have a similar algorithm as for fuzzy $\mathcal{ALC}$ [9], and an optimization in the search space is to use binary search algorithm reducing in that way the satisfiability checks required. Let $\text{glb}(\Sigma, \alpha) \in N^\Sigma$ and $\text{lub}(\Sigma, \alpha) \in 1 - N^\Sigma$, where $N^\Sigma = \{0, 0.5, 1\} \cup \{l : \langle \alpha \geq c \rangle \in \Sigma\} \cup \{1 - l : \langle \alpha \leq l \rangle \in \Sigma\}$ and $1 - N^\Sigma = \{(1 - l) : l \in N^\Sigma\}$. We employ the binary search algorithm, assuming that 0.5 is the middle if we sort the elements of N^Σ. We evaluate if $\Sigma \models (x : C) \geq 0.5$ is satisfied, in case it is we move on to higher degree until $\Sigma \not\models (x : C) \geq n$, indicating that the previous degree is the $\text{glb}(\Sigma, (x : C))$.

Let us assume that $\mathcal{L} = \langle l, \preceq \rangle$ is a linear order, N^Σ is ordered by ascending linear order, and $\|N^\Sigma\|$ is the numbers of lattice degrees. A simple iterative approximation algorithm computing $\text{glb}(\Sigma, \alpha)$ is described in Algorithm 1. The similar algorithm can be easily constructed for the lub.

The described optimizations in Algorithm 1 for glb are very effective since they reduce the search space of tableau independently of Σ used. Based on the previous partition based on connectivity optimization, we could select a partition $\mathcal{A}'$ where the individual of assertion α is contained. Thus a new set of membership degrees only for it with $N^{\mathcal{A}'} \subseteq N^\Sigma$ is evaluated, resulting to greatly less satisfiability checks.

Proposition 14. *By using a binary search on N^Σ, the glb(Σ, α) problem in L-$\mathcal{SHIN}$ with respect to witnessed models can be decided in $\mathcal{O}(\log |N^\Sigma|)$.*

Proof. The complexity of ordering and determining N^Σ are $\mathcal{O}(|N^\Sigma| \cdot \log |\Sigma|)$ and $\mathcal{O}(\Sigma)$, respectively. Hence, the value of $\text{glb}(\Sigma, \alpha)$ can be determined in at most $\mathcal{O}(\log |N^\Sigma|)$ fuzzy entailment tests.

Moreover, the performance of calculating glb can be further improved by preventing recurrent expansion of the expansion rules in $\mathcal{T}$. When $\mathcal{F}$ in which no expansion rule can apply, which is then added to the cached completion-forest. □

5. The Optimized Algorithm

5.1. Algorithm Overview. Our optimized algorithm builds a completion forest $\mathcal{F}$ consisting of a collection of trees whose distinguished roots are arbitrarily connected by edges. We firstly explain the structure of $\mathcal{F}$ in more detail.

Definition 15. A node v in the completion forest $\mathcal{F} = \langle V, E \rangle$ extends an and/or node which is a record with extra two attributes: $sts_v \in \wp^?$, $\text{kind}_v \in I^?$, where $\wp$:= {`unexpanded`, `expanded`, sat, `unsat`},
I:= {`and-node`, `or-node`}

The status of a non-leaf-node is computed from the status of its *R-successor* using its kind (`and-node/or-node`) and treating satisfiability with respect to the TBox $\mathcal{T}$ (i.e., sat) as true and unsatisfiability with respect to the $\mathcal{T}$ (i.e. unsat) as false. When a node gets status `sat` or `unsat`, the status can be propagated to its *R-predecessor* in a way appropriate to the forest and/or structure if desired.

An `and-node` (resp., `or-node`) becomes `unsat` (resp., `sat`) if one of its *R-successors* becomes `unsat` (resp., `sat`). For the remaining case, an `and-node` (resp., `or-node`) becomes `sat` (resp., `unsat`) if the number of its *R-successors* with *status* $\neq$ `sat` (resp., $\neq$ `unsat`) reduces to 0.

It is well known that different orders of expanding nondeterministic rules can result in huge (up to several orders of magnitude) differences in reasoning performance [24]. According to empirical analysis in [24], we have the definition as below.

Definition 16. In our terminology, the following ordering would be optimal for extending: the $\sqcup$-rule has the lowest

Input: A TBox $\mathcal{T}$ in NNF and a finite set of concepts X in NNF
Output: an and-or forest $G = \langle V, E\rangle$, with $\sigma \in V$ as the initial node such that $\sigma.status = \{\texttt{sat, unsat}\}$
begin
create a new node σ with $\Sigma(\sigma) := X \cup \mathcal{T}$ and $\sigma.status = \texttt{unexpanded}$;
let $V := \{\sigma\}$, $E := \emptyset$;
(a) **while** v.status $\notin \{\texttt{sat, unsat}\}$ **do**
(b) choose an node $v \in V$, $v.status := \texttt{unexpanded}$;
if no L-$\mathcal{SHIN}$-tableau rule is applicable to $\Sigma(v)$ **then**
$v.status = \{\texttt{sat }\}$;
else if $\perp$ is applicable to $\Sigma(v)$ **then**
$v.status = \{\texttt{unsat}\}$;
else if γ ($\gamma \in \gamma_1$) is applicable to $\Sigma(v)$ giving concept Y **then**
$v.kind$:= `and-node`, $\Theta := \{Y\}$;
else if γ ($\gamma \in \gamma_2$) is applicable to $\Sigma(v)$ giving concepts Y_1 and Y_2 **then**
$v.kind$:= `or-node`, $\Theta := \{Y_1, Y_2\}$;
else
(i) $v.kind$:= `and-node`;
(ii) for every $\exists R \cdot C \in \mathcal{L}(v)$, apply γ ($\gamma \in \gamma_3$) to $\Sigma(v)$ giving concept $\texttt{trans}_R(\Sigma(v), R) \cup \{C\} \cup \mathcal{T}$ and add this concept to Θ;
(c) **for** $\forall Y \in \Theta$ **do**
(i) **if** *some* $w \in V$ has $\Sigma(w) = Y$ **then**
then add edge (v, w) to E;
(ii) let w be a new node, set $\Sigma(w) := Y$, $w.status$:= `unexpanded`, add w to V, and add edge (v, w) to E;
(d) **if** (v.kind = `or-node` and one of the successors of v has status `sat`) or (v.kind = `and-node` and all the successors of v have status `sat`) **then**
$v.status$:= `sat`, propagate(v);
else if (v.kind = `and-node` and one of the successors of v has status `unsat`) ***or*** (v.kind = `or-node` and all the successors of v have status `unsat`) **then**
$v.status$:= `unsat`, propagate(v);
else
$v.status$:= `expanded`;
(e) **if** $v.status$ = `unsat` ***and*** v in $\mathcal{T}$, $\{\kappa, \neg\kappa\} \in \Sigma(v)$ **then**
dep(κ, v) $\cup$ dep($\neg\kappa, v$);
else if $v.status$:= `unexpanded` **then**
if $\Sigma(v)$ is of the form $\kappa_1 \vee \kappa_2$ **then**
save(T_s);
add κ_1 to $\Sigma(v)$ with dep(κ_1, v) = $\{b\} \cup$ dep($\Sigma(v), v$);
(f) **if** $b \in S$ **then**
add κ_2 to $\Sigma(v)$ with dep(κ_2, v) = $b \cup d$;
else
break;
(g) **If** $\sigma.status$ = `unsat` **then**
return false;
else
return true;

Algorithm 2: A optimized decision procedure for checking satisfiability in L-$\mathcal{SHIN}$.

priority, the generating rules (like the $\geqslant$-rule and the $\exists$-rule) have the second-lowest priority, all other rules except for $\leqslant_*$ have the second-highest priority, and the other rules have the highest priority.

Instead of a top-down approach to most systems used for ordering expansion rules, we use a new ordering heuristic rule by an *OrderedRule* list to control the application of the expansion rules. The fundamental idea is that expansion rules may become applicable whenever a concept is added to a node label, then the node/concept pair is added to the *OrderedRule* list.

Definition 17. Granting the demand of Algorithm 2, the tableau expansion rules in Table 3 are divided into three parts $\{\gamma_1, \gamma_2, \gamma_3\}$:

$\gamma_1 := \{\sqcap_\triangleright, \sqcap_\triangleleft, \forall_\triangleleft, \forall_\triangleright, \forall_+\}$,

$\gamma_2 := \{\leq_\triangleleft, \leq_\triangleleft, \leq_\triangleright, \geq_\triangleleft, \leq_{r\triangleright}, \geq_{r\triangleleft}, \neg_\bowtie, \sqcup_\triangleleft, \sqcap_\triangleright\}$,

$\gamma_3 := \{\exists_\triangleright, \exists_\triangleleft, \exists_+\}$.

Additionally, we use a *OrderedRule* list to control the application of the expansion rules $\{\gamma_1, \gamma_2, \gamma_3\}$. The *OrderedRule* list sorts all these entries according to the descending order queued above and gives access to the first element in the list. The strategy for rule applications given above is the standard one, by applying γ_1 and γ_2 rules as much as possible and then applying γ_3 rules to realise existential concepts. Besides, the strategy for choosing which node in the γ_i to expand next is in the descending order, and that means that, for example, $\sqcap_\triangleright$ in γ_1 will be the first applied rule. The set Θ contains the contents of the resulting $\Sigma(v)$. If the applied tableau rule is in γ_1 , then v has one formula in Θ; if the applied rule is in γ_2 then v has two formulae in Θ; otherwise, $\exists_*$ supply one appropriate formula for Θ.

5.2. The Optimized Algorithm Design. In the following, assume v is current expanded node, b is a nonnegative integer indicating the bth-$\vee$ rule application in the run of the tableau algorithm, $\mathrm{dep}(\varsigma, v)$ is a formula ς in the label of a node v with a dependency set if a node v contains both $\{\kappa, \neg\kappa\} \in \Sigma(v)$, Y is finite sets of concepts, and the set Λ contains the contents of the resulting Y.

To check whether a given finite set X is satisfiable in $\mathcal{T}$, where $\mathcal{T}$ is a Box, the content of the initial node σ with status unexpanded is $X \cup T$. Initially, the queue of nodes V set covers initially only root node σ but can grow as the algorithm proceeds. Next, we present an Exptime decision procedure for L-$\mathcal{SHIN}$ to create a completion forest and/or structure.

We comment on Algorithm 2 given in pseudocode, which determines whether a concept $X \in \mathcal{A}$ is satisfiable with respect to a TBox $\mathcal{T}$, both in NNF. The main while-loop (at line (a)) continues processing nodes until the status of σ is determined to be in $\{\mathtt{sat}, \mathtt{unsat}\}$, or until every node is expanded, whichever happens first.

Inside the main loop, we choose an unexpanded node (at line (b)) and try to apply one and only one of the tableau rules in the order γ_1, γ_2, γ_3 to the current node v. If the applied tableau rule is within γ_1 , then v has one Y; or, if the applied rule is within γ_2, then v has Y_1, Y_2; otherwise, apply $(\exists_\triangleright)$ and $(\exists_\triangleleft)$ to $\Sigma(v)$ giving concept $\mathtt{trans}_R(\Sigma(v), R) \cup \{C\} \cup \mathcal{T}$ and add this concept to Θ.

After that, for every Y in Θ (at line (c)), we create the required successor in the $\mathcal{F}$ only if it does not yet exist. Then, the procedure attempts to compute the status of such a non-leaf-node v employing the kind (`or-node/and-node`) of v and the status of the *R-successor* of v, regarding `unsat` as irrevocably false and `sat` as irrevocably true.

On the condition that previous two steps cannot determine the status of v as `sat or unsat`, then its status is set to be expanded. On the other side, if so (at line (d)), this information is itself propagated to *R-predecessor* of v in the $\mathcal{F}$ via the routine *propagate*(v).

Sequentially, if v.*status* = `unsat` and v in $\mathcal{T}$ (at line (e)), $\{\kappa, \neg\kappa\} \in \Sigma(v)$, then the closure dependency set $S := \mathrm{dep}(\kappa, v) \cup \mathrm{dep}(\neg\kappa, v)$, and the algorithm backtracks to the nth branching point b to apply bth-$*$-rule application. $\mathcal{T}$ would be restored to its state prior to adding κ_1 to $\Sigma(v)$, and the rule would be applied again such that κ_2 was added to $\Sigma(v)$.

Using backjumping, the algorithm will return without expanding the completion trees obtained by adding the various κ_2 to $\Sigma(v)$ immediately (at line (f)) because $b \notin S$. Satisfiable will eventually return $\emptyset$, concluding that X is unsatisfiable.

If σ.*status* = `unsat` (at line (g)), the result is *false*, otherwise *true*.

Proposition 18. *Let $\mathcal{F}$ be constructed by Algorithm 2 for $\Sigma(v)$ and Γ, with σ as the initial node. Then $\Sigma(v)$ is satisfiable with respect to Γ if and only if σ.status* = `sat`.

Lemma 19. *Algorithm 2 assumes that $\Sigma(v)$ is consistent and constructs a model of Γ that satisfies $\Sigma(v)$. Let $\mathcal{F} = \langle V, E\rangle$ be the completion forest constructed by the algorithm for $\Sigma(v) \cup \Gamma$. Suppose the set of concepts carried by the node $v \in V$ is $\hbar(v)$, for every $v \in V$, if v.status* = `unsat`, *then $\hbar(v)$ is inconsistent with respect to Γ.*

Proof (sketch). Since v only depends on its successors and by copying nodes to ensure that the resulting structure is a completion-tree tableau, we can construct a closed tableau with respect to Γ for the set of concepts carried by the node v by induction. □

5.3. Computational Complexity of the Optimized Algorithm. We proceed to analyse the runtime of the global caching algorithm, under suitable sanity assumptions on the set of modal rules. Specifically, in order to ensure that executions of the algorithm run in exponential time, we need to assume n is the size of input, that is, the sum of the size of $\Sigma(v) \cup \Gamma$. This construction uses both *caching* and *backjumping* techniques.

Proposition 20. *Algorithm 2 runs in ExpTime.*

Proof (sketch). Since $\hbar(v) \subseteq S(\Sigma(v) \cup \Gamma)$, so the set contains $2^{\mathcal{O}(n)}$ concepts. Omitting the execution time of procedure *propagate*, every $v \in V$ is expanded only once and every expansion takes $2^{\mathcal{O}(n)}$ time units.

When v.*status* becomes `sat` or `unsat`, the *propagate* procedure executes $2^{\mathcal{O}(n)}$ basic steps directly involved with v, so the total time of the executions of *propagate* is of rank $2^{2\cdot\mathcal{O}(n)}$.

Using backjumping, the algorithm returns from bth branching point immediately because $b \notin S$. This is obviously true for all of the preceding branching points, so all calls to b-th branching point will return without expanding the completion trees to $\Sigma(v)$. So that the size of $\Sigma(v) \cup \Gamma$ can be further reduced. □

6. Evaluation

In this section, we provide test results about the usefulness of the optimizations of our algorithm on the test from revised LogicsWork Bench (LWB) benchmarks [25] and the DL98 benchmarks (http://dL.kr.org/dl98/comparison). All the experiments were conducted under Ubuntu 12.04 on a PC with 2.40 GHz Intel Core 2 CPU Q8300 and 2 GB RAM, using the 1s time limit for each test problem. We consider only the number of solved problems instead of the used CPU time.

6.1. Programming Language. The problem we are dealing with in this work is whether the optimized tableau algorithm proposed by us can be implemented to give an efficient prover for $L\text{-}\mathcal{SHIN}$. This problem can be raised as what optimization techniques can be implemented to obtain a prover for $L\text{-}\mathcal{SHIN}$ that is both optimal from the theoretical point of view and efficient in practice. To the best of our knowledge, there is no DL/uncertainty prover/reasoner for $L\text{-}\mathcal{SHIN}$. Thus we have developed in C++ a tableau prover for $L\text{-}\mathcal{SHIN}$, which uses serial of optimization methods described above.

C++ is chosen to implement tableau prover because of its increase efficiency, portability, and extendibility. Pointers are intensively used for the data structures of the tableau prover to reduce memory use and to allow caching objects like formulas, sets of formulas, and nodes of a completion forest. Advanced data structures of C++ with low complexity like the *templates set, map, multimap, and priority queue* are also intensively used. And it is more convenient to extending it for other description logics in a modular way [26].

6.2. Experimental Results on Usefulness of the Optimizations. Because neither do good benchmarks for $L\text{-}\mathcal{SHIN}$ seem to exist, so we found it hard to compile a good set of test problems. Although the Tableaux'98 and random test suites could show how optimisations perform on propositional modal logics, neither is very good for our purposes.

We therefore recast the K, KT, and S4 problem sets from the LogicsWork Bench (LWB) benchmarks and the `T98-sat` problem set from the DL98 benchmarks (there are 4 sets of DL98 test suite, and only *concept satisfiability tests are employed*) to be compatible with the fuzzy logic $L\text{-}\mathcal{SHIN}$ over finite lattices. Due to the space limit, we omit the detailed process of transformation. The three anterior benchmarks measure the performance of the system when testing the satisfiability of large concept expressions without reference to a Tbox. Moreover, DL98 TBox problem set texttttT98-sat test is added to contain TBox axioms. The correctness of the system is also tested by checking that the answer is as expected.

Our algorithm is fairly detailed, so a naive implementation should be straightforward. The preprocessing optimisations ("partition based on connectivity" and "dealting with TBox axioms") described in Section 4.1 take at most polynomial time to run and reduce the search space more significantly than the extended framework of Donini and Massacci [27]. So it is sensible to use them in almost every case in our tableau prover. After preprocessing, it is crucial to employ core satisfiability optimisations presented in Section 4.2 that try to improve the performance of the core satisfiability tester.

We intend to show which of the optimisations are most effective, so we have had compile-time configuration options included that can be used to turn on and off or vary the previous optimisations. We chose to test various combinations with all the other optimisations enabled and then to test the same optimisations options, with each of the optimisations turned off one by one. The optimisations that were removed are as follows.

(i) *ALL*: turn on all the optimizations.

(ii) *No partition based on connectivity (NP)*: turn off ABox partition.

(iii) *No dealing with TBox axioms (ND)*: turn off dealing with TBox axioms.

(iv) *No backjumping (NB)*: turn off backjumping.

(v) *No global caching (NC)*: turn off global caching.

We end up with 5 configurations and ran each of these configurations over the nine test suites from satisfiability checking problems. For the corresponding subset of tests and the corresponding set of options, each table cell includes the number of solved problems, where the diamond symbol denotes 21 (the number of test problems in the subset). For each configuration, there are five entries corresponding to timeouts of 1 second, 2 seconds, 4 seconds, 8 seconds, and 16 seconds.

6.2.1. K($\mathcal{L}$) Tests. Table 4 obviously reveals that, amongst the tested configurations, turning on all the optimizations provides the most effective method. These two other optimisations are the most effective, whose absence in the experiments has made them unacceptably slow. For k_lin_n($\mathcal{L}$) and k_lin_p($\mathcal{L}$), the methods of global caching significantly outperform backjumping. There is a nice distinction between the two preprocessing optimisations methods. Partition based on connectivity and dealing with TBox axioms is less important at least within our set of test cases. For k_lin_n($\mathcal{L}$), k_path_n($\mathcal{L}$), and k_path_p($\mathcal{L}$), "dealing with TBox axioms" significantly outperforms "partition based on connectivity," and for the other problems, "partition based on connectivity" outperforms "dealing with TBox axioms."

6.2.2. KT($\mathcal{L}$) Tests. Similar testing was performed on KT($\mathcal{L}$) tests, and results are showed in Table 5. Again, there is too much data to present it that global caching significantly outperforms other optimization methods in both of its incarnations. Particularly, we have undergone an increase of memory usage by a worst-case factor of two in case of backjumping in comparison to no backjumping. This time, for the most problems, the "partition based on connectivity" method is on par with "dealing with TBox axioms."

6.2.3. S4($\mathcal{L}$) Tests. The results of S4($\mathcal{L}$) tests, given in Table 6, show that backjumping performs very well compared

Table 4: Results for K($\mathscr{L}$) tests.

Problem	ALL	NP	ND	NB	NC
k_branch_n($\mathscr{L}$)	18 ◇ ◇ ◇ ◇	15 16 18 19 ◇	17 20 ◇ ◇ ◇	14 16 19 20 ◇	7 9 10 12 15
k_branch_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	17 18 20 ◇ ◇	18 20 ◇ ◇ ◇	17 19 ◇ ◇ ◇	8 10 10 13 16
k_d4_n($\mathscr{L}$)	10 11 12 14 14	5 6 6 7 7	7 7 8 8 9	4 4 5 6 6	3 4 4 5 6
k_d4_p($\mathscr{L}$)	17 20 ◇ ◇ ◇	6 6 7 7 7	8 8 9 9 9	3 4 5 6 8	4 4 4 5 5
k_dum_n($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	20 ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇
k_dum_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇	18 ◇ ◇ ◇ ◇
k_grz_n($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
k_grz_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	20 ◇ ◇ ◇ ◇
k_lin_n($\mathscr{L}$)	19 ◇ ◇ ◇ ◇	6 8 10 13 17	4 7 9 13 16	16 18 20 ◇ ◇	15 17 19 20 ◇
k_lin_p($\mathscr{L}$)	6 8 10 12 15	6 8 11 14 16	7 10 13 16 19	17 19 ◇ ◇ ◇	15 17 19 20 ◇
k_path_n($\mathscr{L}$)	11 11 13 14 14	5 5 6 6 7	4 4 4 5 6	3 3 3 4 4	2 2 3 3 3
k_path_p($\mathscr{L}$)	12 12 13 14 15	5 5 6 7 8	4 4 4 4 6	3 3 4 4 5	3 3 3 3 4
k_ph_n($\mathscr{L}$)	8 8 9 10 10	8 8 8 8 9	8 8 8 9 10	8 8 8 8 8	7 7 7 7 8
k_ph_p($\mathscr{L}$)	8 8 9 9 10	7 8 8 8 8	8 8 9 9 10	7 8 8 8 8	7 7 7 7 8
k_ploy_n($\mathscr{L}$)	20 ◇ ◇ ◇ ◇	17 18 20 ◇ ◇	18 20 ◇ ◇ ◇	11 12 13 14 16	8 9 9 10 11
k_ploy_p($\mathscr{L}$)	19 ◇ ◇ ◇ ◇	18 19 ◇ ◇ ◇	19 20 ◇ ◇ ◇	11 12 14 15 17	9 10 11 12 14
k_t4p_n($\mathscr{L}$)	20 ◇ ◇ ◇ ◇	11 12 14 16 17	12 14 16 17 19	6 7 8 8 10	4 5 5 6 7
k_t4p_p($\mathscr{L}$)	20 ◇ ◇ ◇ ◇	11 13 15 17 18	12 15 17 18 20	6 7 9 10 12	4 4 5 5 6

Table 5: Results for KT($\mathscr{L}$) tests.

Problem	ALL	NP	ND	NB	NC
kt_branch_n($\mathscr{L}$)	10 11 12 14 14	5 6 6 7 7	7 7 8 8 9	4 4 5 6 6	3 4 4 5 6
kt_branch_p($\mathscr{L}$)	17 20 ◇ ◇ ◇	6 6 7 7 7	8 8 9 9 9	3 4 5 6 8	4 4 4 5 5
kt_45_n($\mathscr{L}$)	8 9 10 11 12	8 9 10 11 12	8 9 10 11 12	8 9 10 11 13	8 9 10 11 12
kt_45_p($\mathscr{L}$)	17 18 19 20 ◇	17 18 19 20 ◇	17 18 19 20 ◇	17 18 19 20 ◇	17 18 19 20 ◇
k_dum_n($\mathscr{L}$)	20 ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇	17 19 ◇ ◇ ◇	13 15 16 18 19
k_dum_p($\mathscr{L}$)	20 ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇	19 ◇ ◇ ◇ ◇	18 20 ◇ ◇ ◇	19 ◇ ◇ ◇ ◇
k_grz_n($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
k_grz_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
k_md_n($\mathscr{L}$)	8 8 8 9 9	8 8 8 8 9	7 7 7 7 8	6 6 7 7 7	4 4 4 4 5
k_md_p($\mathscr{L}$)	7 7 7 7 8	8 8 9 9 9	6 7 7 7 7	6 7 7 7 7	3 3 3 4 4
k_path_n($\mathscr{L}$)	4 4 4 5 5	4 4 4 5 5	4 4 4 4 5	3 4 4 4 4	3 3 3 4 4
k_path_p($\mathscr{L}$)	4 5 5 5 6	4 4 5 5 5	4 4 4 5 5	3 3 4 4 4	3 3 4 4 4
k_ph_n($\mathscr{L}$)	7 7 8 8 9	6 7 7 8 8	7 7 8 8 8	6 6 7 7 8	5 5 6 6 7
k_ph_p($\mathscr{L}$)	6 7 7 8 8	6 6 7 7 8	6 6 7 7 8	6 7 7 8 8	5 6 6 7 7
k_ploy_n($\mathscr{L}$)	4 4 5 6 6	4 4 5 5 6	4 5 5 5 6	3 3 5 5 5	2 2 4 4 4
k_ploy_p($\mathscr{L}$)	8 8 10 10 11	8 8 10 10 10	8 8 9 10 10	6 6 9 9 10	4 4 6 6 7
k_t4p_n($\mathscr{L}$)	3 3 5 5 6	3 3 4 5 6	3 3 4 4 6	2 2 3 4 4	1 1 3 3 3
k_t4p_p($\mathscr{L}$)	5 5 7 7 7	5 5 6 7 7	5 5 7 7 7	1 2 2 4 4	1 1 2 2 3

to other methods. This result is surprising; global caching is less effective at almost all in this test suite. The probable reason for the ineffectiveness of the method is that, with such a small number of literals in the successor nodes, the purely propositional problems can always be worked out deterministically, while the performance is contingent on the efficiency of propositional reasoning at the root node.

There appears to be no clear winner between the two methods "partition based on connectivity" and "dealing with TBox axioms." "Dealing with TBox axioms" method outperforms the "partition based on connectivity" method for most problems, and for the problems s4_ipc_p($\mathscr{L}$) and s4_path_p($\mathscr{L}$), "dealing with TBox axioms" method beats the "partition based on connectivity" method. However, the differences are not great.

6.2.4. K($\mathscr{L}$) TBox Tests. Due to latticed LWB benchmarks not covering TBox problem, so test on the test set DL'98

Table 6: Results for S4($\mathscr{L}$) tests.

Problem	ALL	NP	ND	NB	NC
s4_branch_n($\mathscr{L}$)	10 11 12 13 15	9 10 11 12 14	8 9 10 11 12	8 9 10 12 13	8 9 10 11 13
s4_branch_p($\mathscr{L}$)	12 13 14 16 17	11 12 13 14 15	10 11 12 13 14	10 11 13 14 16	10 11 13 14 16
s4_45_n($\mathscr{L}$)	12 15 19 ◇ ◇	10 13 16 20 ◇	9 12 15 18 ◇	6 9 11 14 17	7 9 12 15 18
s4_45_p($\mathscr{L}$)	15 19 ◇ ◇ ◇	13 16 19 ◇ ◇	12 15 18 20 ◇	9 12 16 20 ◇	10 13 17 20 ◇
s4_ipc_n($\mathscr{L}$)	9 11 12 15 16	11 13 15 16 18	10 12 14 16 17	10 12 12 14 15	10 12 12 15 15
s4_ipc_p($\mathscr{L}$)	16 19 ◇ ◇ ◇	14 16 18 19 ◇	15 17 19 20 ◇	18 ◇ ◇ ◇ ◇	17 20 ◇ ◇ ◇
s4_grz_n($\mathscr{L}$)	18 ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
s4_grz_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
s4_md_n($\mathscr{L}$)	8 8 9 9 10	7 8 8 9 9	6 6 7 7 8	5 6 6 7 7	6 6 7 7 8
s4_md_p($\mathscr{L}$)	7 7 8 8 9	6 6 7 8 8	6 7 7 8 8	4 4 5 5 5	5 5 6 6 7
s4_path_n($\mathscr{L}$)	3 3 3 3 4	2 3 3 3 3	2 3 3 3 3	3 4 4 4 4	3 3 3 3 3
s4_path_p($\mathscr{L}$)	4 4 4 4 4	3 3 4 4 4	3 4 4 4 4	4 4 5 5 5	3 4 4 4 4
k_ph_n($\mathscr{L}$)	9 9 9 10 10	8 9 9 9 10	8 8 9 9 10	8 8 8 9 9	10 10 10 10 11
k_ph_p($\mathscr{L}$)	11 11 11 11 12	10 10 10 10 11	9 10 10 10 10	7 8 8 8 8	12 12 12 13 13
s4_s5_n($\mathscr{L}$)	7 7 8 8 8	7 7 7 8 8	7 7 7 7 8	6 7 7 7 7	6 6 6 6 7
s4_s5_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	20 ◇ ◇ ◇ ◇
s4_t4p_n($\mathscr{L}$)	3 4 4 6 7	3 3 4 6 7	3 3 4 6 6	2 2 3 3 4	1 1 3 3 4
s4_t4p_p($\mathscr{L}$)	5 7 7 9 10	5 7 8 9 10	5 7 8 8 10	4 4 6 6 8	3 4 4 6 6

Table 7: Results for K($\mathscr{L}$) TBox tests.

Problem	ALL	NP	ND	NB	NC
k_branch_n($\mathscr{L}$)	5 5 6 6 7	5 5 5 6 7	4 5 5 6 6	4 4 5 5 6	5 5 6 6 7
k_branch_p($\mathscr{L}$)	8 8 9 9 10	8 8 9 9 9	7 7 8 8 9	6 7 7 8 8	8 8 9 9 10
k_d4_n($\mathscr{L}$)	4 5 5 5 6	4 4 5 5 6	3 3 4 5 6	3 3 4 4 5	4 5 5 5 5
k_d4_p($\mathscr{L}$)	16 18 19 20 ◇	15 18 19 20 20	13 15 17 18 18	14 14 15 16 17	13 15 16 17 18
k_dum_n($\mathscr{L}$)	17 19 20 ◇ ◇	17 18 20 ◇ ◇	10 12 15 17 20	8 10 13 16 19	16 18 19 20 ◇
k_dum_p($\mathscr{L}$)	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇	20 ◇ ◇ ◇ ◇	◇ ◇ ◇ ◇ ◇
k_grz_n($\mathscr{L}$)	13 15 17 19 ◇	12 14 16 18 ◇	11 13 14 17 20	10 12 14 17 19	13 15 17 18 20
k_grz_p($\mathscr{L}$)	6 9 11 14 16	5 8 10 13 15	5 7 9 11 15	4 6 9 11 13	4 7 10 12 14
k_lin_n($\mathscr{L}$)	10 10 11 11 12	10 10 10 11 11	9 10 10 10 11	9 9 10 10 10	9 10 10 11 11
k_lin_p($\mathscr{L}$)	7 7 7 8 8	7 7 7 7 8	6 6 7 7 8	6 6 7 7 7	6 6 7 7 7
k_path_n($\mathscr{L}$)	12 12 13 14 14	11 12 13 13 14	6 6 9 10 10	5 5 7 7 7	11 12 13 13 14
k_path_p($\mathscr{L}$)	14 14 15 15 16	13 14 14 15 15 11 11	9 9 11 12 12	7 7 8 9 9	10 10 11
k_ph_n($\mathscr{L}$)	10 10 13 13 15	9 10 12 12 15	9 10 13 13 15	6 7 7 8 8	10 10 12 13 14
k_ph_p($\mathscr{L}$)	12 12 16 16 19	11 12 15 16 18	10 11 14 14 14	8 8 10 10 11	10 10 12 12 13
k_ploy_n($\mathscr{L}$)	16 19 ◇ ◇ ◇	17 18 20 ◇ ◇	12 16 20 ◇ ◇	11 15 19 20 ◇	12 16 20 ◇ ◇
k_ploy_p($\mathscr{L}$)	18 ◇ ◇ ◇ ◇	17 20 ◇ ◇ ◇	15 18 ◇ ◇ ◇	14 17 20 ◇ ◇	14 17 19 ◇ ◇
k_t4p_n($\mathscr{L}$)	7 9 11 14 17	7 9 11 15 17	6 8 10 13 15	4 7 9 13 16	5 7 9 12 15
k_t4p_p($\mathscr{L}$)	10 13 15 17 20	10 12 15 17 19	7 11 14 16 18	6 10 13 15 18	6 10 13 15 17

T98 kb is employed. As shown in Table 7, in this test, "dealing with TBox axioms" is by far the most important optimisation. The next most important optimisations are global caching and backjumping. That is because of explosion in the large size of the search space (2^{kn}) in the problems, which results in a destructive degradation in performance, even when optimisations such as global caching and backjumping are used. "Partition based on connectivity" method is almost totally ineffective.

The results reveal that, amongst the tested configurations, global caching is more efficient over the other methods in most but not all cases. Since global caching severely prunes the search space by avoiding multiple expansions of the same node, by enabling greater propagation of `sat` and `unsat` via subset checking and hence allowing more cutoffs to be done [28]. Backjumping approach can be applied together with global caching well, which reduces the number of tested nondeterministic branches largely especially for S4($\mathscr{L}$)

Tests, while in particular, the method may increase memory usage. When considering TBox problem, "dealing with TBox axioms" is the most important optimisation. Sometimes "partition based on connectivity" method is very effective but not all the time.

As shown in the test results, the difference between using preprocessing optimizations or not is less significant than the core satisfiability optimization at least within our set of test cases. Furthermore, preprocessing optimizations cooperates very well with core satisfiability optimization techniques in most of the time.

7. Conclusion and Future Work

Many papers offer technical contributions but without practical motivations. Our method for L-$\mathcal{SHIN}$ is definitely promising and should be extended to more expressive logics to test whether it remains feasible. In the further way, the tableau algorithm can be used on the large scale data set such as linked data to solve the practical problem. As far as we know L-$\mathcal{SHIN}$ is the first presentation of a reasoning algorithm for such complex fuzzy DL languages over finite residuated lattices. Besides, this is the first time that optimized techniques were used to tableau reasoning for completion forest in DLs augmented by De Morgan lattices. This enabled us to treat the case of general concept inclusions instead of the simpler acyclic TBoxes.

Of course, there remains plenty of work to do with the tableau algorithm of L-$\mathcal{SHIN}$, such as to directly and efficiently handle general TBoxes and to deal with concrete domains. We intend to extend L-$\mathcal{SHIN}$ also for more description logics under more semantics. Besides, the presented dependency management allows for more informed backjumping, while also supporting the creation of precise cache unsatisfiability entries [29]. Among others, better heuristics are needed to guide the choices needed during tableau reasoning, and better use of cached information is expected to provide significant benefits. Additionally, like other related research areas such as Semantic Web, most DL reasoners are implemented in Java for usability by many applications, so the Java implementation of the optimized algorithm will be carried out in the future work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] F. Bobillo and U. Straccia, "Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic *SROIQ*," *Information Sciences*, vol. 181, no. 4, pp. 758–778, 2011.

[2] U. Straccia, "Description logics over lattices," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 14, no. 1, pp. 1–16, 2006.

[3] S. Borgwardt and R. Peñaloza, "Description logics over lattices with multi-valued ontologies," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, vol. 2, pp. 768–773, AAAI Press, 2011.

[4] S. Borgwardt and R. Peñaloza, "Fuzzy ontologies over lattices with t-norms," in *Proceedings of the 24th International Workshop on Description Logics (DL '11)*, CEUR Electronic Workshop Proceedings, 2011.

[5] S. Borgwardt and R. Peñaloza, "A tableau algorithm for fuzzy description logics over residuated de morgan lattices," in *Proceedings of the 6th International Conference on Web Reasoning and Rule Systems (RR '12)*, M. Krötzsch and U. Straccia, Eds., vol. 7497 of *Lecture Notes in Computer Science*, pp. 9–24, Springer, 2012.

[6] S. Borgwardt and R. Peñaloza, "Finite lattices do not make reasoning in $\mathcal{ALC}$ harder," in *URSW*, pp. 51–62, 2011.

[7] P. Hájek, "Making fuzzy description logic more general," *Fuzzy Sets and Systems*, vol. 154, no. 1, pp. 1–15, 2005.

[8] P. Hájek, "On witnessed models in fuzzy logic. III. Witnessed Gödel logics," *Mathematical Logic Quarterly*, vol. 56, no. 2, pp. 171–174, 2010.

[9] U. Straccia, "Reasoning within fuzzy description logics," *Journal of Artificial Intelligence Research*, vol. 14, pp. 137–166, 2001.

[10] G. Stoilos, G. Stamou, J. Pan, N. Simou, and V. Tzouvaras, "Reasoning with the fuzzy description logic f-$\mathcal{SHIN}$: theory, practice and applications," in *Uncertainty Reasoning For the Semantic Web I*, pp. 262–281, 2008.

[11] G. Stoilos, G. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks, "Reasoning with very expressive fuzzy description logics," *Journal of Artificial Intelligence Research*, vol. 30, no. 5, pp. 273–320, 2007.

[12] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.

[13] I. Horrocks, U. Sattler, and S. Tobies, "Reasoning with individuals for the description logic SHIQ," in *Proceedings of the 17th International Conference on Automated Deduction (CADE '00)*, D. McAllester, Ed., vol. 1831 of *Lecture Notes in Computer Science*, pp. 482–496, Springer, 2000.

[14] S. Tobies, "Complexity results and practical algorithms for logics in knowledge representation," *Journal of Artificial Intelligence Research*, vol. 12, pp. 199–217, 2000.

[15] F. Bou, M. Cerami, and F. Esteva, "Finite-valued lukasiewicz moda logic is pspace-complete," in *Proceedings of the 22nd International Joint Conference On Artificial Intelligence (IJCAI '11)*, vol. 2, pp. 774–779, AAAI Press, 2011.

[16] V. Haarslev, H. I. Pai, and N. Shiri, *Optimizing Tableau Reasoning in Alc Extended with Uncertainty*, 2008.

[17] N. Simou, T. Mailis, G. Stoilos, and G. Stamou, "Optimization techniques for fuzzy description logics," in *Proceedings of the International Workshop on Description Logics (DL '10)*, pp. 244–254, 2010.

[18] D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider, "Optimizing terminological reasoning for expressive description logics," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 277–316, 2007.

[19] D. Tsarkov and I. Horrocks, "Efficient reasoning with range and domain constraints," in *Proceedings of the Description Logic Workshop (DL '04)*, vol. 104, pp. 41–50, 2004.

[20] I. Horrocks, U. Hustadt, U. Sattler, and R. Schmidt, "Computational modal logic," in *Handbook of Modal Logic*, P. Blackburn, J. van Benthem, and F. Wolter, Eds., chapter 4, pp. 181–245, Elsevier, 2006.

[21] A. B. Baker, *Intelligent backtracking on constraint satisfaction problems: experimental and theoretical results [Ph.D. thesis]*,

University of Oregon, Eugene, Ore, USA, 1995, UMI Order no. GAX95-29049.

[22] F. Oppacher and E. Suen, "HARP: a tableau-based theorem prover," *Journal of Automated Reasoning*, vol. 4, no. 1, pp. 69–100, 1988.

[23] R. Goré and L. Postniece, "An experimental evaluation of global caching for $\mathscr{ALC}$ (system description)," in *Automated Reasoning*, pp. 299–305, 2008.

[24] D. Tsarkov and I. Horrocks, "Ordering heuristics for description logic reasoning," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 609–614, 2005.

[25] P. Balsiger, A. Heuerding, and S. Schwendimann, "A benchmark method for the propositional modal logics K, KT, S4," *Journal of Automated Reasoning*, vol. 24, no. 3, pp. 297–317, 2000.

[26] L. A. Nguyen, "An efficient tableau prover using global caching for the description logic $\mathscr{ALC}$," *Fundamenta Informaticae*, vol. 93, no. 1–3, pp. 273–288, 2009.

[27] F. M. Donini and F. Massacci, "EXPtime tableaux for $\mathscr{ALC}$," *Artificial Intelligence*, vol. 124, no. 1, pp. 87–138, 2000.

[28] R. Goré and L. A. Nguyen, "Optimised exptime tableaux for $\mathscr{ALC}$ using sound global caching, propagation and cutoffs," 2007, http://www.mimuw.edu.pl/~nguyen/papers.

[29] A. Steigmiller, T. Liebig, and B. Glimm, "Extended caching, backjumping and merging for expressive description logics," in *Automated Reasoning*, vol. 7364 of *Lecture Notes in Computer Science*, pp. 514–529, Springer, Heidelberg, Germany, 2012.