Journal of Southeast University (English Edition) Vol. 22, No. 4, pp. 496 – 500 Dec. 2006 ISSN 1003—7985

New reasoning algorithm based on EFALC

Zhou Bo[1] Lu Jianjiang[1] Zhang Yafei[1] Kang Dazhou[2] Li Yanhui[2]

([1]Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

([2]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: The current extended fuzzy description logics lack reasoning algorithms with TBoxes. The problem of the satisfiability of the extended fuzzy description logic EFALC cut concepts w. r. t. TBoxes is proposed, and a reasoning algorithm is given. This algorithm is designed in the style of tableau algorithms, which is usually used in classical description logics. The transformation rules and the process of this algorithm is described and optimized with three main techniques: recursive procedure call, branch cutting and introducing sets of mesne results. The optimized algorithm is proved sound, complete and with an EXPTime complexity, and the satisfiability problem is EXPTime-complete.

Key words: extended fuzzy description logic; cut concept; TBox; reasoning algorithm

Description logics (DLs)[1] are a class of knowledge representation languages with well-defined semantics and determinable reasoning methods. Web applications often need to represent fuzzy knowledge, especially when dealing with text, multimedia or uncertain data. However, classical DLs are insufficient to deal with such fuzzy knowledge. Therefore, it is necessary to add fuzzy features to description logics.

Straccia presented a fuzzy extension of typical ALC(FALC)[2], and gave a constraint propagation calculus for reasoning. FALC just offers limited but not sufficient expressive power of complex fuzzy information. Some discussion about reducing ALC into classical ALC was given in Ref. [3]; however, the reduction did not extend the expressive abilities of FALC. To overcome its insufficiency, we present a new family of extended fuzzy description logics (EFDLs), in which cut sets of fuzzy concepts and fuzzy roles are considered as the atomic concepts and atomic roles[4-5]. Some complexity results and reasoning techniques in EFDLs were proposed in Refs. [6 – 7], and in these papers, expressive advantages of our framework were discussed in detail. However, these current extended fuzzy description logics lack reasoning algorithms with TBoxes. This paper discusses the reasoning problems for the extended fuzzy description logic EFALC with TBoxes, proves the complexity of the problems is EXPTime-complete, and gives an optimized reasoning algorithm, which is proved to be complete, sound and with a worst complexity in EXPTime.

Received 2006-04-25.

Foundation items: The National Natural Science Foundation of China (No. 60403016), the Weaponry Equipment Foundation of PLA Equipment Ministry (No. 51406020105JB8103).

Biographies: Zhou Bo (1982—), male, graduate; Lu Jianjiang (corresponding author), male, doctor, associate professor, jjlu@seu.edu.cn.

1 Reasoning Algorithm for EFALC with Tboxes

The definitions of syntax, semantics, reasoning tasks, and reasoning properties are given for EFALC in Refs. [4 – 5]. In this paper, we mainly consider satisfiability of EFALC cut concepts w. r. t. TBoxes: A given cut concept $C_{[n_1, \ldots, n_k]}$ is satisfiable w. r. t. TBox T, iff there is an interpretation I such that I is a model of T and $C^I_{[n_1, \ldots, n_k]} \neq \varnothing$[5].

1.1 Transformation rules

In this section, we will propose a reasoning algorithm for satisfiability of EFALC cut concepts w. r. t. TBoxes. This algorithm is developed in the style of tableau algorithms, which are usually used in classical DLs. For satisfiability of ALC concept C_0, the tableau algorithm starts with an ABox $A = \{x_0 : C_0\}$, then extends A with transformation rules and tries to create a complete and clash-free ABox, which can be directly converted into an interpretation satisfying C_0. For satisfiability of EFALC cut concepts, we modify a similar but more complex algorithm. Given a cut concept $C_{0[n_1, \ldots, n_k]}$, our algorithm starts with $A = \{x_0 : C_{0[n_1, \ldots, n_k]}\}$, then uses transformation rules to extend A, and tries to create a complete and clash-free ABox.

The transformation rules are given as follows, and these rules can be applied to an individual x in ABox A w. r. t. TBox T, when x is not blocked, and A is not closed:

- $\cap$-rule

  Condition: x: $D_{[n_i, \ldots, n_s]} \cap E_{[n_{s+1}, \ldots, n_l]} \in A$, x: $D_{[n_i, \ldots, n_s]}$ or x: $E_{[n_{s+1}, \ldots, n_l]} \notin A$.

  Action: $A^1 = A \cup \{x: D_{[n_i, \ldots, n_s]}, x: E_{[n_{s+1}, \ldots, n_l]}\}$.

- $\cup$-rule

  Condition: x: $D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_l]} \in A$, x: $D_{[n_i, \ldots, n_s]}$, $E_{[n_{s+1}, \ldots, n_l]} \notin A$.

  Action: $A^1 = A \cup \{x: D_{[n_i, \ldots, n_s]}\}$; $A^2 = A \cup \{x: E_{[n_{s+1}, \ldots, n_l]}\}$.

- $\exists$-rule

  Condition: x: $\exists R_{[n_i]}. D_{[n_{i+1}, \ldots, n_l]} \in A$, $\forall y \in A$, $(x, y): R_{[n_i]}$ or y: $D_{[n_{i+1}, \ldots, n_l]} \notin A$.

  Action: $A^1 = A \cup \{z: D_{[n_{i+1}, \ldots, n_l]}, (x, z): R_{[n_i]}\}$, where z is newly-generated.

- $\forall$-rule

  Condition: x: $\forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_l]}$, $(x, y): R_{[n_j]} \in A$, y: $D_{[n_{i+1}, \ldots, n_l]} \notin A$, $n_i \leqslant n_j$.

  Action: $A^1 = A \cup \{y: D_{[n_{i+1}, \ldots, n_l]}\}$.

- $\neg$-rule

  Condition: x: $\neg B_{[n_i]}$, x: $B_{[n_j]} \in A$, $n_i + n_j \geqslant 1$.

  Action: $A^1 = A$, A^1 is announced closed.

- KB-rule

  Condition: x: $C_T \notin A$.

  Action: $A^1 = A \cup \{x: C_T\}$.

Definition 1 Here are the definitions of some terms used in transformation rules.

1) For any individual x in A, the label of x is $L(x) = \{C_{[n_i, \ldots, n_j]} \mid x: C_{[n_i, \ldots, n_j]} \in A\}$. If $\exists$-rule is applied to x and a new individual y is created, $(x, y): R_{[n]}$, then y is a successor of x. Let descendant be transitive closure of a successor. For any individuals x, y, if x is a descendant of y, and $L(x) \subseteq L(y)$, then x is blocked by y, and all descendants of x are also blocked.

2) In EFALC, an ABox A is closed, iff $\{x: \neg B_{[n_i]}, x: B_{[n_j]} (n_i + n_j \geqslant 1)\} \subseteq A$.

3) In order to consider the TBox T, a cut concept C_T is introduced to represent the semantics of T. $C_T \overset{\text{def}}{=} \cap \{\neg C_{[n_1, \ldots, n_i]} \cup D_{[m_1, \ldots, m_j]} \mid C_{[n_1, \ldots, n_i]} \subseteq D_{[m_1, \ldots, m_j]} \in T\}$. An interpretation I is a model of T iff $\Delta^I = (C_T)^I$, viz. each element in the domain belongs to $(C_T)^I$.

If none of the transformation rules can be applied to A, then A is complete. An ABox A is closed, iff it is announced closed, or else A is open. A finite set of ABoxes S is complete iff any ABox A in S is complete. S is open iff $\exists A \in S, A$ is open.

Starting from $S_0 = \{A_0: = \{x_0: C_{0[n_1, \ldots, n_k]}\}\}$, apply the transformation rules to current S_i exhaustively. We denote S_{i+1} is the subsequence set of ABoxes of S_i, so there is a chain of S_i by the application of rules: $S_0 \rightarrow S_1 \rightarrow \ldots \rightarrow S_i \rightarrow \ldots \rightarrow S_n$. If the complete set S_n is open, return that the cut concept $C_{0[n_1, \ldots, n_k]}$ is satisfiable. However, when applying the transformation rules without optimization, the worst complexity is 2NEXPTime (it can be proved analogous to lemma 1), so an optimized algorithm is needed.

1.2 An optimized algorithm

Classic ALC has EXPTime algorithms for satisfiability of concepts w. r. t. TBoxes[8]. We adopt the optimizations in that algorithm of ALC to design an optimized algorithm for the satisfiability of FALC cut concepts w. r. t. TBoxes. Algorithm 1 adopts three main techniques for optimization: first, it applies the rules in a depth-first manner by calling a recursive procedure; secondly, it optimizes the $\cup$-rule based on the idea that for any $C_{[n_1, \ldots, n_h]} \cup D_{[n_{h+1}, \ldots, n_k]}$, if $C_{[n_1, \ldots, n_h]}$ is found to be satisfiable, then $C_{[n_1, \ldots, n_h]} \cup D_{[n_{h+1}, \ldots, n_k]}$ is satisfiable, and it is not necessary to check whether $D_{[n_{h+1}, \ldots, n_k]}$ is satisfiable; thirdly, two sets U and V are used to record the labels causing the closed ABox and labels whose individuals cannot be applied by rules to avoid repeatedly checking and whose corresponding ABoxes are open. Algorithm 1 and the procedure Sat are shown in the following.

Algorithm 1 Algorithm for satisfiability of EFALC cut concepts w. r. t. TBoxes

Input: EFALC cut concept $C_{0[n_1, \ldots, n_k]}$ and TBox T.

Output: $C_{0[n_1, \ldots, n_k]}$ is satisfiable or not satisfiable.

Process: Let $U \leftarrow \varnothing$, $V \leftarrow \varnothing$.

If Sat(x_0, $\{x_0: C_{0[n_1, \ldots, n_k]}\}$) is true then return that $C_{0[n_1, \ldots, n_k]}$ is satisfiable;

else return that $C_{0[n_1, \ldots, n_k]}$ is unsatisfiable.

Procedure Sat

Input: individual x, ABox A.

Output: true or false.

Process: ① If $L(x) \in U$ then return false;

② If $L(x) \in V$ then return true;

③ If any transformation rule can be applied to x in A w. r. t. T then apply a rule, do case

KB, $\cap$, $\neg$: $A \leftarrow A_1$;

if A is closed then goto ⑤;

$\forall$: $A \leftarrow A_1$, goto ③;

$\cup$: if Sat(x, A_{21}) is true then $A \leftarrow A_{21}$, goto ⑥;

if Sat(x, A_{22}) is true then $A \leftarrow A_{22}$, goto ⑥;

else goto ⑤;

$\exists$: $A \leftarrow A_1$, goto ③;

④ If all rules cannot be applied to x, x is not blocked and x has m successors $z_1, \ldots, z_m$

then do

for $i = 1$ to m do

if Sat(z_i, A) is false then goto ⑤;

else goto ⑥;

⑤ $U \leftarrow U \cup \{L(x)\}$, return false;

⑥ $V \leftarrow V \cup \{L(x)\}$, return true.

In ③ of procedure Sat, for A, we denote the resultant ABox(es) by A_1 when applying $\neg$, $\cap$, $\exists$, $\forall$ or KB-rule; by A_{21}, A_{22} when applying $\cup$-rule.

2 Algorithm Analysis

Definition 2 For any cut concept $C_{[n_i, \ldots, n_j]}$, the size of $C_{[n_i, \ldots, n_j]}$ ($\| C_{[n_i, \ldots, n_j]} \|$) is the number of symbols used to write down C, where every name in N_I, N_C, N_R and every constructor ($\neg$, $\cap$, $\cap$, $\exists$, $\forall$) is written in one symbol.

The set of subconcepts of $C_{[n_i, \ldots, n_j]}$ is defined as

$$\mathrm{sub}(C_{[n_i, \ldots, n_j]}) \overset{\mathrm{def}}{=} \{C_{[n_i, \ldots, n_j]}\} \cup \begin{cases} \mathrm{sub}(D_{[n_i, \ldots, n_j]}) & \text{case (a)} \\ \mathrm{sub}(D_{[n_{i+1}, \ldots, n_j]}) & \text{case (b)} \\ \mathrm{sub}(D_{[n_i, \ldots, n_l]}) \cup \mathrm{sub}(E_{[n_{l+1}, \ldots, n_j]}) & \text{case (c)} \end{cases}$$

when $C_{[n_i, \ldots, n_j]}$ is of the form (a) $\neg D_{[n_i, \ldots, n_j]}$, (b) $\exists R_{[n_i]}.D_{[n_{i+1}, \ldots, n_j]}$ or $\forall R_{[n_i]}.D_{[n_{i+1}, \ldots, n_j]}$, (c) $D_{[n_i, \ldots, n_l]} \cap E_{[n_{l+1}, \ldots, n_j]}$ or $D_{[n_i, \ldots, n_l]} \cup E_{[n_{l+1}, \ldots, n_j]}$. It follows $|\mathrm{sub}(C_{[n_i, \ldots, n_j]})| \leqslant \| C_{[n_i, \ldots, n_j]} \|$.

Lemma 1 (termination) Algorithm 1 is terminated.

Proof The input size of the algorithm is $N = \| C_{0[n_1, \ldots, n_k]} \| + \| C_T \|$. From transformation rules, for any label L, it follows $L \subseteq \mathrm{sub}(C_{0[n_1, \ldots, n_k]}) \cup \mathrm{sub}(C_T)$. So $|L| \leqslant N$, and there are at most 2^N different labels.

For any ABox A, its individuals can form a tree T_A. Let x_0 be the root node. For any node x in T_A, the successors of x are the child-nodes of x in T_A. Since there are at most 2^N different labels, when the length of a branch in T_A is more than 2^N, there must be x, y on the branch such that $L(x) = L(y)$, then the leaf node is blocked. So the length of any branch in T_A is no more than $2^N + 1$. No individual can have more than N successors. Therefore T_A has at most $N(2^N + 1)$ individuals.

The execution of any Sat(x, A) is a call, x is its individual. If a new call is created in Sat(x, A), then the new call is a child of Sat(x, A). All calls can form a tree T_P: let Sat(x_0, $\{x_0 : C_{0[n_1, \ldots, n_k]}\}$) be the root node, for any node Sat(x, A) in T_P, its children are its child-nodes in T_P.

For any Sat(x, A), its child must satisfy that the label is increased or the individual is a successor of x. So any call can be executed in EXPTime if the cost of its children is not considered. So the depth of T_P is no more than $N(2^N + 1)$. For any call in T_P, it can only apply no more than N^2 transformation rules; and its children are no more than $\max(2, N) : 2$ in ③, N in ④. So there are at most $(\max(2, N))^{N(2^N+1)}$ calls in T_P. Therefore, algorithm 1 is terminated in 2NEXPTime.

Furthermore, by introducing U and V, its worst complexity is EXPTime. From ① to ⑥, V is always increasing, so we can ensure that the numbers of total calls are exponential.

Lemma 2 (complexity) The complexity of algorithm 1 is EXPTime.

Proof In the tree T_P, for any two calls Sat(x, A_1) and Sat(y, A_2), they have the same label L, i. e., $L(x) = L(y)$, and Sat(x, A_1) is created before Sat(y, A_2).

If Sat(y, A_2) is a descendant of Sat(x, A_1), then y must be a descendant of x in A_2 and $L(y) \subseteq L(x)$. So y is blocked and Sat(y, A_2) has no child.

If Sat(y, A_2) is not a descendant of Sat(x, A_1), then Sat(x, A_1) ends before Sat(y, A_2) is created. From ⑥, we obtain $L(x) \in V$, and Sat(y, A_2) has no child.

Therefore, for any nodes with the same label in T_P, only one of them can have children. There are at most 2^N different labels. All nodes in T_P, except the root, are children of other nodes. Since any node has no more than $\max(2, N)$ direct children, there are at most $1 + \max(2, N)\, 2^N$ nodes in T_P, i. e., the total number of calls are exponential to N. In lemma 1, it proves that every call can be executed in EXPTime. So algorithm 1 can be executed in EXPTime.

Lemma 3 (completeness) For any given $n_i \in [0, 1]$, $C_{0[n_1, \ldots, n_k]}$ is unsatisfiable w. r. t. T if algorithm 1 returns that $C_{0[n_1, \ldots, n_k]}$ is unsatisfiable, i. e., Sat(x_0, $\{x_0 : C_{0[n_1, \ldots, n_k]}\}$) returns false.

Proof For any label L, there exists a model of T, I, such that $\cap \{(C_{[n_i, \ldots, n_j]})^I \mid C_{[n_i, \ldots, n_j]} \in L\} \neq \varnothing$, then we say I is a model of L. If L has no model, we say L is unsatisfiable. From ⑤, $L \in U$ iff there exists a call with label L returning false. It can be proved that for any $L \in U$, L is unsatisfiable.

When $|U| = 0$, for any $L \in U$, L is unsatisfiable.

When $|U| = 1$, the foremost call adding L into U must be in ③ case $\neg$, and there is $\{\neg B_{[n_i]}, B_{[n_j]}\} \subseteq L$ and $n_i + n_j \geqslant 1$. Then, there is no interpretation I such that $(\neg B_{[n_i]})^I \cap (B_{[n_j]})^I \neq \varnothing$. So L is unsatisfiable. Assume that when $|U| = r$, for any $L \in U$, L is unsatisfiable.

When $|U| = r + 1$, let Sat(x, A) add a new L into U. It may

1) apply $\neg$-rule and there is $\{\neg B_{[n_i]}, B_{[n_j]}\} \subseteq L$, $n_i + n_j \geqslant 1$. The proof is the same as the above.

2) apply $\neg$, KB, $\cap$-rule: let the label of the child be L_0. And $L_0 \in U$, L_0 is unsatisfiable. So L is unsatisfiable.

- In case $\neg$, $L = L_0$.
- In case KB, assume that L has a model I. Then $\Delta^I = (C_T)^I$; since $L_0 = L \cup \{C_T\}$, I is also a model of L_0. It is in contradiction to that L_0 is unsatisfiable. So L is unsatisfiable.
- The case $\cap$ is similar to the KB case.

3) apply $\cup$, $\forall$-rule: let the labels of the two children be L_1 and L_2. $L_1, L_2 \in U$, and L_1, L_2 are unsatisfiable. So L is also unsatisfiable.

- In case $\forall$, $L = L_1 = L_2$.
- In case $\cup$, there is $D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_j]} \in L$, $L_1 = L \cup \{D_{[n_i, \ldots, n_s]}\}$ and $L_2 = L \cup \{E_{[n_{s+1}, \ldots, n_j]}\}$. Assume that L has a model I. Then $(D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_j]})^I \neq \varnothing$. From the interpretation definition[5], $(D_{[n_i, \ldots, n_s]})^I \neq \varnothing$ or $(E_{[n_{s+1}, \ldots, n_j]})^I \neq \varnothing$. So I is a model of L_1 or L_2. It is a contradiction.

4) have a child from ④: let the label of the child be $L_0 \in U$, and L_0 is unsatisfiable. L_0 is created by applying $\exists$, $\forall$-rule to L. So L is also unsatisfiable. Here only prove the $\forall$-rule case, other are similar.

From transformation rules, if the $\forall$-rule is applied, then x: $\forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_l]} \in A$ and there is a y with (x, y): $R_{[n_j]} \in A$, $n_i \leqslant n_j$. Assume that L has a model I. Then $x \in (\forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_l]})^I$, $(x, y) \in (R_{[n_j]})^I$. From $n_i \leqslant n_j$, $(x, y) \in (R_{[n_i]})^I$. Then x satisfies that $\forall y \in \Delta^I$, $(x, y) \in (R_{[n_i]})^I \rightarrow y \in (D_{[n_{i+1}, \ldots, n_l]})^I$. So y satisfies that $y \in (D_{[n_{i+1}, \ldots, n_l]})^I$. The $\forall$-rule only adds $D_{[n_{i+1}, \ldots, n_l]}$ into y's label L_0. So I is a model of L_0. It is a contradiction.

Therefore, for any $r \geqslant 0$, when $|U| = r$, it follows for any $L \in U$, L is unsatisfiable. If Sat(x_0, $\{x_0$: $C_{0[n_1, \ldots, n_k]}\}$) returns false, then $\{C_{0[n_1, \ldots, n_k]}\} \in U$, so $C_{0[n_1, \ldots, n_k]}$ is unsatisfiable.

Lemma 4(soundness) For any given $n_i \in [0, 1]$, if algorithm 1 returns that $C_{0[n_1, \ldots, n_k]}$ is satisfiable, it can build a model of T, I, such that $(C_{0[n_1, \ldots, n_k]})^I \neq \varnothing$, i. e., $C_{0[n_1, \ldots, n_k]}$ is satisfiable w. r. t. T.

Proof We can build an interpretation $I = \langle \Delta^I, \cdot^I \rangle$ from ABox A created by the call Sat(x_0, $\{x_0$: $C_{0[n_1, \ldots, n_k]}\}$): Δ^I is the set of all individuals in A which is not blocked. For any $x \in \Delta^I$, $B \in N_C$, $B^I(x) = \max(\{0\} \cup \{n_i \mid x: B_{[n_i]} \in A\})$.

For any $x, y \in \Delta^I$, $R \in N_R$, $R^I(x, y) = \max(\{0\} \cup \{n_i \mid (x, y) \in \pi R_{[n_i]}\})$, where $\pi R_{[n_i]} = \{(x, y) \mid$ there exists z such that (x, z): $R_{[n_j]} \in A$, $n_i \leqslant n_j$, where z is blocked by y or $z = y\}$.

For any $x, y \in \Delta^I$ and $R_{[n_i]}$, if (x, y): $R_{[n_i]} \in A$, then $R^I(x, y) \geqslant n_i$, $(x, y) \in (R_{[n_i]})^I = \pi R_{[n_i]}$. So I satisfies all the role facts in A. In succession, prove that for any $x \in \Delta^I$ and $C_{[n_i, \ldots, n_j]}$, if x: $C_{[n_i, \ldots, n_j]} \in A$, then $x \in (C_{[n_i, \ldots, n_j]})^I$.

When $\| C_{[n_i, \ldots, n_j]} \| = 1$, $C_{[n_i, \ldots, n_j]}$ must be of the form $B_{[n_i]}$. If x: $B_{[n_i]} \in A$, then $B^I(x) \geqslant n_i$, $x \in (B_{[n_i]})^I$. Assume that for any $x \in \Delta^I$ and $\| C_{[n_i, \ldots, n_j]} \| \leqslant r$, if x: $C_{[n_i, \ldots, n_j]} \in A$, then $x \in (C_{[n_i, \ldots, n_j]})^I$.

When $\| C_{[n_i, \ldots, n_j]} \| = r + 1$, $C_{[n_i, \ldots, n_j]}$ must be of the form, i. e., $\exists R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]}$, $\forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]}$, $D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_j]}$ or $D_{[n_i, \ldots, n_s]} \cap E_{[n_{s+1}, \ldots, n_j]}$:

1) If $C_{[n_i, \ldots, n_j]} = D_{[n_i, \ldots, n_s]} \cap E_{[n_{s+1}, \ldots, n_j]}$, then from the $\cap$-rule, it follows x: $D_{[n_i, \ldots, n_s]} \in A$ and x: $E_{[n_{s+1}, \ldots, n_j]} \in A$. So x: $(D_{[n_i, \ldots, n_s]})^I$ and x: $(E_{[n_{s+1}, \ldots, n_j]})^I$. Then it has x: $(D_{[n_i, \ldots, n_s]})^I \cap (E_{[n_{s+1}, \ldots, n_j]})^I = (D_{[n_i, \ldots, n_s]} \cap E_{[n_{s+1}, \ldots, n_j]})^I$.

2) If $C_{[n_i, \ldots, n_j]} = D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_j]}$ or $\exists R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]}$, these two cases are similar to case 1). It has x: $(D_{[n_i, \ldots, n_s]} \cup E_{[n_{s+1}, \ldots, n_j]})^I$ or $x \in (\exists R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]})^I$.

3) If $C_{[n_i, \ldots, n_j]} = \forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]}$, then for any $(x, y) \in (R_{[n_i]})^I = \pi R_{[n_i]}$, from the $\forall$-rule, for any z such that (x, z): $R_{[n_j]} \in A$ and $n_i \leqslant n_j$, z: $D_{[n_{i+1}, \ldots, n_j]} \in A$; for any y such that $y = z$ or y blocks z, i. e., $(x, y) \in \pi R_{[n_i]}$, it has $L(z) \subseteq L(y)$, y: $D_{[n_{i+1}, \ldots, n_j]} \in A$; then, since $\| D_{[n_{i+1}, \ldots, n_j]} \| \leqslant r$, $y \in (D_{[n_{i+1}, \ldots, n_j]})^I$. From the interpretation definition, $x \in (\forall R_{[n_i]}. D_{[n_{i+1}, \ldots, n_j]})^I$.

Since x_0 cannot be blocked, $x_0 \in \Delta^I$; x_0: $C_{0[n_1, \ldots, n_k]} \in A$, so $x_0 \in (C_{0[n_1, \ldots, n_k]})^I$. From the KB-rule, for any

$x \in \Delta^I$, x: $C_T \in A$, so $x \in (C_T)^I = \Delta^I$. Therefore, I is a model of T and satisfies $(C_{0[n_1,\ldots,n_k]})^I \neq \varnothing$, i. e., $C_{0[n_1,\ldots,n_k]}$ is satisfiable w. r. t. T.

Any classical ALC knowledge base K can be transformed to an equivalent EFALC knowledge base K'. The transformation from classical to fuzzy is trivial: for any concept name B and role name R in K, replace them by $B_{[1]}$ and $R_{[1]}$ in K'. Since the satisfiability of ALC concepts w. r. t. TBoxes is EXPTime-complete, and it can be polynomially reduced to the satisfiability of EFALC cut concepts w. r. t. TBoxes, the latter is EXPTime-hard but has an EXPTime algorithm (algorithm 1). It follows:

Theorem 1 The satisfiability of EFALC cut concepts w. r. t. TBoxes is EXPTime-complete.

3 Conclusion

We propose a reasoning algorithm for the extended fuzzy description logic EFALC. The algorithm is for satisfiability of EFALC cut concepts w. r. t. TBoxes. In detail, we describe the transformation rules and the process of this algorithm. We also prove that it is sound, complete, and with an EXPTime complexity, and the satisfiability problem is EXPTime-complete. Further work includes extending EFALC with more concept and role constructs and considering TBoxes with variables.

References

[1] Baader F, Calvanese D, McGuinness D L, et al. *The description logic handbook*: *theory*, *implementation*, *and applications* [M]. Cambridge: Cambridge University Press, 2003.

[2] Straccia U. Reasoning within fuzzy description logics [J]. *Journal of Artificial Intelligence Research*, 2001, **14**(1): 137 - 166.

[3] Straccia U. Transforming fuzzy description logics into classical description logics [A]. In: *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* [C]. Lisbon, 2004. 385 - 399.

[4] Lu J J, Xu B W, Li Y H, et al. A family of extended fuzzy description logics [J]. *International Journal of Business Intelligence and Data Mining*, 2006, **1**(4): 384 - 400.

[5] Li Y H, Xu B W, Lu J J, et al. An extended fuzzy description logic [J]. *Journal of Southeast University (Natural Science Edition)*, 2005, **35**(5): 683 - 687. (in Chinese)

[6] Li Y H, Xu B W, Lu J J, et al. On the computational complexity of the extended fuzzy description logic with numerical constraints [J]. *Journal of Software*, 2006, **17**(5): 968 - 975. (in Chinese)

[7] Li Y H, Lu J J, Xu B W, et al. A fuzzy extension of description logic ALCH [A]. In: *Lecture Notes in Artificial Intelligence* [C]. Springer, 2005, **3789**: 152 - 161.

[8] Donini F M, Massacci F. EXPTIME tableaux for ALC [J]. *Artificial Intelligence*, 2000, **124**(1): 87 - 138.

一个新的基于 EFALC 的推理算法

周 波[1] 陆建江[1] 张亚非[1] 康达周[2] 李言辉[2]

([1] 解放军理工大学指挥自动化学院,南京 210007)

([2] 东南大学计算机科学与工程学院,南京 210096)

摘要:针对目前的扩展模糊描述逻辑缺乏 TBox 约束下的推理算法,提出 TBox 约束下扩展模糊描述逻辑 EFALC 截概念可满足性问题及其推理算法. 该算法的设计参考用于经典描述逻辑的 tableau 算法;详细描述了 EFALC 的转化规则和推理算法的过程,并使用递归函数调用、分支裁减和引入中间结果集合这 3 种技术对算法进行优化. 证明了该优化推理算法的最坏时间复杂性为指数,且具有完备性和正确性,同时证明了 TBox 约束下 EFALC 截概念可满足性问题的复杂性是指数时间完全的.

关键词:扩展模糊描述逻辑;截概念;TBox;推理算法

中图分类号:TP311. 13