Tableau Caching for Description Logics with Inverse and Transitive Roles

Yu Ding and Volker Haarslev

Concordia University, Montreal, Quebec, Canada
{ding_yu|haarslev}@cse.concordia.ca

Abstract. Modern description logic (DL) reasoners are known to be less efficient for DLs with inverse roles. The current loss of performance is largely due to the missing applicability of some well-known optimization techniques, especially the one for caching the satisfiability status of modal successors. In this paper, we present a rule synthesis technique from which an estimation of the potential back-propagation of constraints can be made. This estimation can be applied to both the concept classifier and the satisfiability tester. This paper presents a tableau caching technique for $\mathcal{SHI}$ as a first step to improving the performance of tableau-based DL reasoners for logics offering the use of inverse roles. The proposed techniques underwent a first empirical evaluation with a prototype DL reasoner for $\mathcal{SHI}$ using a set of synthetically generated knowledge bases. The initial results indicate an significant improvement in runtime performance once caching is effectively enabled.

1 Motivation

Description logics (DLs) are a family of logic based formalisms for terminological knowledge representation and reasoning [15]. Descending from KL-ONE [1], they now enjoy a wide spectrum of applications. Behind this success is a long line of research and implementation efforts in DL reasoners. Most modern DL systems are based on tableau algorithms. Such algorithms are first introduced in [2] by Schmidt-Schauss and Smolka. In spite of the high worst case complexity of the satisfiability problem [15] for most expressive DLs (typically ExpTime-complete), highly optimized implementations have been shown to work well in many realistic applications [3, 9, 14].

DLs express unary relations in terms of concepts and binary relations in terms of roles. To describe inverse binary relationships, inverse roles are a necessary language element. One of the first approaches to allow the declaration of inverse roles was in the early nineties when some DL systems started to support very expressive DLs with inverse roles [5]. Later, the theoretical result on elimination of inverse roles based on equi-satisfiability was established [7].

Tableau-based DL systems typically employ a wide range of optimization techniques, most of which were designed and established unfortunately without appropriate consideration of inverse roles (which would be reasonable if the elimination of inverse roles had become realistic in applications). If directly applied to DLs with inverse roles, some well-known optimizations become less efficient or even invalid (due to the two-way propagation of universal restrictions introduced by inverse roles). The caching technique is one such example that suddenly turns unsound when inverse roles are considered. The current technique of inverse role elimination could introduce an overwhelming number of GCIs (General Concept Inclusions) that are difficult for tableau algorithms. Other attempts were tried, e.g., the *dynamic blocking* technique [8] which can be considered as a successful attempt in adapting the *blocking* [4] technique to DLs with inverse roles.

The literature addressing the issues caused by inverse role is not plentiful, and a systematic treatment is still to be done. This paper will discuss the soundness problem of common caching techniques for DLs with inverse roles. We extend our previous study [16] to a more expressive description logic, i.e., $\mathcal{SHI}$, a logic with expressive role constructs (i.e., transitive roles, inverse roles, and role hierarchies). We assume the reader's familiarity with the DL $\mathcal{SHI}$ and tableau algorithms. We introduce the rule synthesis approach, discuss the use of heuristics for concept classification, and show how to apply the same technique to obtain sound tableau caching.

2 Synthesizing Rules

In this section, we present our rule synthesis approach in terms of a reachability analysis of the underlying unfolding rules. Our reachability analysis is based on a multi-graph $G = (V, E)$ constructed from the syntactic structure of the given unfolding rules of $\mathcal{T}$. The initial reachability relation is represented as ϵ-edges and its transitive closure can be computed according to a set of reachability extension rules that are presented in the first subsection. The potential back propagations is contained in this closure. After this, we show how the estimated potential back propagation can be used to obtain sound sub-tableau caching as well as potential subsumers, respectively.

2.1 Potential Back Propagation

Due to the interaction of universal restrictions with inverse roles, constraints can propagate up and down the tableau tree. We refer to this upward propagation of constraints as *back propagation*. Here we present a way to synthesize the unfolding rules [15] to determine *potential back propagation* (PBP). We view PBP a reflexive and transitive relation, and formulate it as *reachability* in a directed graph (containing multi-edges and cycles). In the discourse, we use the notation $G = (V, E)$ for a directed multi-graph, where V denotes a set of nodes and E denotes a set of edges (binary relations) for pairs of nodes in V.

We say C (also D) is *non-modally used* by $C \sqcap D$ and $C \sqcup D$; and say $\forall R.(A \sqcup B)$ (resp. $\exists R.(A \sqcup B)$) *modally uses* A (also B) through $\forall R$ (resp. $\exists R$). Note we are not using the transitive *use relation*, for details please see [13].

Definition 1. *Given a $\mathcal{SHI}$ expression in* `NNF` *(negation normal form), its reachability graph is a multi-graph $G = (V, E)$ formed according to a function f such that: (1) for every concept literal C, there is a node $f(C) \in V$; (2) for every $\mathcal{SHI}$ (sub-)expression s, there is a unique node $f(s) \in V$; (3) if expression s_1 non-modally uses s_2, then $\epsilon \in E(f(s_1), f(s_2))$; if s_1 modally uses s_2 through $\otimes R$ for some role name R, then $\otimes R \in E(f(s_1), f(s_2))$, where $\otimes$ stands for the connective $\exists$ or $\forall$.*

Recursively an expression could be decomposed into concept literals and correspondingly G is constructed.

Definition 2. *Given $\mathcal{T}$, a $\mathcal{SHI}$ TBox of unfolding rules each of which is of the form $C_L \sqsubseteq D$, where C_L is a concept literal and D is in* `NNF`*, a reachability graph $G = (V, E)$ for $\mathcal{T}$ is formed such that for each rule $C_L \sqsubseteq D$: (1) there is a node $f(C_L) \in V$; (2) there is a node $f(D) \in V$; (3) $\epsilon \in E(f(C_L), f(D))$.*

The initial reachability graph G is built upon the syntactic structure only. A *non-modal use* relationship is represented by an ϵ-edge, a *modal use* relationship is represented by an edge tagged with the modality connective concatenated with the corresponding role name. The unfolding operation is represented by connecting the concept literal with an ϵ-edge to the root of the graph representation of its right-hand side.

In Table 1 three rules are given for extending the reachability relation from the initial $G = (V, E)$. We use E^* to denote the transitive closure of E over ϵ-edges.

$\exists\forall$-rule:	if	there exist nodes x, y, z such that $\exists R_1 \in E^*(x, y)$ and $\forall R_2 \in E^*(y, z)$ and $R_1 \sqsubseteq^* \texttt{Inv}(R_2)$
	then	$E(x, z) := E(x, z) \cup \{\epsilon\}$
$\forall\forall$-rule:	if	there exist nodes x, y, z such that $\forall R_1 \in E^*(x, y)$ and $\forall S \in E^*(y, z)$ (for short let $\texttt{Inv}(S)$ be R_2) and there exists $R_3 \in \mathbf{R}$ s.t. $R_3 \sqsubseteq^* R_i$ for $i \in \{1, 2\}$
	then	$E(x, z) := E(x, z) \cup \{\epsilon\}$
$\forall\exists$-rule:	if	there exist nodes x, y, z such that $\forall R_1 \in E^*(x, y)$ and $\exists R_2 \in E^*(y, z)$ and there exists $\texttt{Trans}(S)$ s.t. $R_2 \sqsubseteq^* S \sqsubseteq^* R_1$
	then	$E(y, y) := E(y, y) \cup \{\forall S\}$

Table 1. Reachability Extension Rules over $\mathcal{R}$ for $\mathcal{SHI}$.

Initially, the reachability graph $G = (V, E)$ is constructed according to all unfolding rules in $\mathcal{T}$. Then, an extension of G is based on a simulation of the $\forall$-rule in tableau algorithms w.r.t. $\mathcal{R}$. For a given TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ in DL $\mathcal{SHI}$, the PBP is contained in the final reachability relation in E^*.

To see how PBP is obtained, we compare the tableau expansion with the reachability extension. In tableau algorithms, both decomposition and selection are done by the $\sqcap$-rule and the $\sqcup$-rule; on the other hand, in the reachability graph, only decomposition is simulated. Also, in tableau algorithms, the firing of $\exists$-rule is subject to conditions, whereas in reachability graph, we make a simplification and assume it can be always fired. Furthermore, in tableau algorithms, the firing of $\forall$-rule depends on the existence of an edge (e.g., might be provided by a sibling expression), but in a reachability graph, we simplify that condition to only consider its predecessors and successors (as if that sibling always exists). In summary, by making these simplification and discarding inessential factors, the reachability graph is constructed in a very conservative way to grasp every potential back propagation. The essence for the reachability graph is that if PBP (ϵ-edge in the final $G = (V, E^*)$) is impossible, then it is guaranteed that no back propagation will take place in the corresponding application of the tableau algorithm.

For a given TBox $\mathcal{T}$ (of a set of unfolding rules) and a role hierarchy $\mathcal{R}$, there is a corresponding reachability graph $G = (V, E^*)$. For a concept literal C, we define a function $\texttt{Reach}(C) = \{y \in V \mid \epsilon \in E^*(f(C), y)\}$. In the following, we use the function $\texttt{Reach}$ without mentioning its $\mathcal{T}$ and $\mathcal{R}$.

2.2 Heuristic for Sound Sub-tableau Caching

Based on $\texttt{Reach}(C)$, we define the function $\texttt{Watch}$ applicable to a concept literal C: $\texttt{Watch}(C) = \bigcup_{x \in \texttt{Reach}(C)} \{\texttt{Inv}(R) \mid \exists y \in V \ s.t. \ \forall R \in E(x, y)\}$, where R is a (inverse) role. For a set of concept literals C_s, we define $\texttt{CWatch}(C_s) = \bigcup_{C \in C_s} \texttt{Watch}(C)$. Let R_s be a set of (inverse) roles, we use $R_s{\downarrow}$ to denote the union of subroles for each $r \in R_s$ w.r.t. role hierarchy $\mathcal{R}$. At this point, we are able to define a boolean function $\texttt{Safe}(C_s, R)$ that can be used to guarantee caching soundness by excluding potential unsound situations: $\texttt{Safe}(C_s, R) = \texttt{true}$ iff $R \notin \texttt{CWatch}(C_s){\downarrow}$. The following lemma follows naturally from the previous analysis.

Lemma 1 (Sound Caching). *Given the TBox $\mathcal{T}$ and the role hierarchy $\mathcal{R}$, let C_s and C_w be two sets of concept literals, R be a (inverse) role, The set C_s can be cached at the position of R-successor if: (1) $C_s \subseteq C_w$; and (2) $\texttt{Safe}(C_s, R)$; and (3) C_w has a model.*

Proof (sketch). (1) Assume C_w has the model $\mathcal{I}_w$. The construction process for $\mathcal{I}_w$ can be used to guide that of C_s, and C_s has a model $\mathcal{I}_s$ inside which each node's label is a subset of that of $\mathcal{I}_w$; (2) The condition $\texttt{Safe}(C_s, R)$ guarantees the root of $\mathcal{I}_s$ has no $\texttt{Inv}(R)$ predecessor, even if $\mathcal{I}_w$ needs a $\texttt{Inv}(R)$ predecessor.

2.3 Heuristic for Potential Subsumer

Classification is the process of computing the most-specific subsumption relationships (i.e., parents and children) of each concept name to other concept names mentioned in a TBox. The parents and children of a certain concept name are computed in the so-called top-down and bottom-up traversal phases, respectively. For large knowledge base, it is particularly important to avoid as many traversal as possible. A typical optimization is to use a heuristics-guided traversal which exploits information about *told subsumers* [3] or *potential subsumer* [12] to restrict the search space.

However, even the better *potential subsumer* technique considers only the transitive *non-modal use* relation [12] among concept names. For DLs with inverse roles, *potential subsumers* could appear in *modal-use* relations. For example, given a simple TBox $\mathcal{T} = \{A \sqsubseteq \exists R.B, B \sqsubseteq \forall R^{-}.C\}$ (note that $\mathcal{T}$ has only primitive definitions and no cyclic rules). The subsumption of $A \sqsubseteq C$ holds, and the subsumer C is neither explicitly told nor non-modal discernible.

For DLs with inverse roles, the *potential subsumer* technique remains useful if we take those concept names appearing in $\texttt{Reach}(A)$ into consideration. Similarly, this estimation is also applicable to related ABox reasoning tasks [15].

3 Discussion and Related Work

As a first step to evaluate the proposed techniques, i.e., caching and back-propagation estimation for DLs with inverse roles, a experimental tableau-based reasoner has been developed in Java. It employs several optimization techniques such as lexical normalization, semantic branching, backjumping (in a weak form), and a optimized version of dynamic blocking (tailored to DLs free of number restrictions). Besides the basic platform, two components were implemented, one for back-propagation estimation, and one for caching (the latter relies on the former). The proposed techniques were then evaluated with a set of synthetically generated satisfiability tests. These preliminary benchmarks indicated a performance gain of a factor of 2 to 10. A more realistic benchmark using RacerPro is in preparation.

The following papers in the literature are relevant in our context to caching and blocking. DeGiacomo et. al. [6] demonstrated for the DL $\mathcal{ALC}$ that the tableau caching technique (especially unsatisfiability caching) is necessary for obtaining a worst case optimal tableau algorithm. For super-set and sub-set caching see [12]. For various blocking techniques for DLs with inverse roles see [11, 8]. Papers which systematically addressed the classification problem include [3, 12]. However, they commonly focused on DLs without inverse roles. Recent papers focusing on the optimization of classification did not address the anomaly when inverse roles are present in the TBox.

Satisfiability tests for DLs with inverse roles are empirically hard for tableau-based reasoning systems. For instance, it is required that the blocking technique (i.e., the cycle detection mechanism) dynamically guarantees soundness. Furthermore, witness nodes are restricted to be ancestor nodes only. This dynamic behavior together with a limited choice of witness nodes generally makes the tableau-based procedures less efficient than necessary. Our preliminary experiments show that the proposed approach can be feasibly implemented as part of a one-pass parsing phase for tableau-based DL reasoners. The availability of the caching technique dramatically speeds up the reasoning process. First, the witness space is no longer restricted to ancestor nodes. Second, the caching technique requires

no re-checking and thus it is safe to discard nodes once they are successfully cached. By doing this, the tableau algorithm is more space-economic. Our approach can successfully deal with a set of (possibly cyclic) unfolding rules and a role hierarchy, thus it well meets the typical requirement from real applications.

4 Conclusion

The common caching techniques are no longer applicable in the presence of inverse roles, since information can be pushed backwards, which might invalidate the cache. Closely related to known techniques of knowledge compilation, we presented in this paper a technique for synthesizing the unfolding rules (of the terminological box) to estimate potential back propagations. This technique is currently applied to terminological knowledge bases with cyclic axioms and expressive roles in the expressive DL $\mathcal{SHI}$. The presented solution computes a reachability graph before testing the satisfiability of a concept in order to determine, when backpropagation is impossible and hence caching is safe. Based on this, a sound tableau caching technique was obtained in a relatively straightforward way. As was expected, better run-time performance was observed in our preliminary experiments due to the use of cache. We also pointed out that the estimation could be used for other purposes such as in locating potential subsumers during concept classification. We are aware that further refinement is possible for the technique presented here.

References

1. Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., Borgida, A.: Living with CLASSIC: When and how to use a KL-ONE-like language. Principles of Semantic Networks, edited by John Sowa, Morgan Kaufmann, (1991) 401-456
2. Schmidt-Schauss M., Smolka G.: Attributive concept descriptions with complements. Artificial Intelligence, Vol 48, (1991) 1-26
3. Baader, F., Hollunder, B., Nebel, B., Profitlich, H.-J., Franconi, E.: An Empirical Analysis of Optimization Techniques for Terminological Representation Systems - or - Making KRIS get a move on: KR-92, (1992) 270-281

4. Buchheit M., Donini F., Nutt W., Schaerf A.: Decidable Reasoning in Terminological Knowledge Representation Systems. Journal of Artificial Intelligence Research, Vol 1 (1993) 109-138.
5. Paolo Bresciani, Enrico Franconi, and Sergio Tessaris: Implementing and testing expressive description logics: a preliminary report. DL-95 (1995).
6. De Giacoma, G., Donini, F., Massacci, F.: EXPTIME Tableaux for ALC: DL-96, (1996)
7. Diego Calvanese, Giuseppe De Giacomo, Riccardo Rosati: A Note on Encoding Inverse Roles and Functional Restrictions in ALC Knowledge Bases. DL-98 (1998).
8. I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. Journal of Logic and Computation, 9(3), (1999) 385-410
9. Ian Horrocks: Using an expressive description logic: FaCT or fiction?: KR-98 (1998)
10. Volker Haarslev, Ralf Möller: Consistency Testing: The RACE Experience: Proc. of TABLEAUX'2000, (2000) 57-61
11. F. Baader, U. Sattler: An Overview of Tableau Algorithms for Description Logics. Studia Logica, 69, (2001) 5-40
12. Volker Haarslev, Ralf Möller: High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study. Proc. of IJCAI'2001 (2001) 161-166
13. Volker Haarslev: Theory and Practice of Visual Languages and Description Logics: Habilitation Thesis, Computer Science Department, University of Hamburg, September (2001)
14. Volker Haarslev, Ralf Möller: RACER System Description: Proc. of IJCAR'2001 (2001)
15. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
16. Yu Ding, Volker Haarslev: Towards Efficient Reasoning for Description Logics with Inverse Roles. DL-05 (2005) 208-215