A C++ reasoner for the description logic $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ [*] (work in progress)

Domenico Cantone, Marianna Nicolosi-Asmundo, and Daniele Francesco Santamaria

University of Catania, Dept. of Mathematics and Computer Science
email: {cantone,nicolosi,santamaria}@dmi.unict.it

Abstract. We present an ongoing implementation of a KE-tableau based reasoner for a decidable fragment of stratified elementary set theory expressing the description logic $\mathcal{DL}\langle\mathsf{4LQS}^{\mathsf{R},\times}\rangle(\mathbf{D})$ (shortly $\mathcal{DL}_{\mathbf{D}}^{4,\times}$). The reasoner checks the consistency of $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-knowledge bases (KBs) represented in set-theoretic terms. It is implemented in C++ and supports $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-KBs serialized in the OWL/XML format.
To the best of our knowledge, this is the first attempt to implement a reasoner for the consistency checking of a description logic that is represented via a fragment of set theory that can also classify standard OWL ontologies.

1 Introduction

Computable set theory is a research field rich of decidability results, however only recently some of its fragments have been applied in the context of knowledge representation and reasoning for the semantic web. Such efforts are motivated by the characteristics of the considered set-theoretic fragments. These provide very expressive and unique formalisms that combine the modelling capabilities of a rule language with the constructs of description logics. In particular, the set-theoretic fragment $\mathsf{4LQS}^{\mathsf{R}}$ [1], involving variables of four sorts, pair terms, and a restricted form of quantification over variables of the first three sorts, is appropriate for these finalities since it turned out to be efficiently implementable.

In [4], $\mathsf{4LQS}^{\mathsf{R}}$-quantifier-free atomic formulae of the types $x = y$, $x \in X^1$, $\langle x, y\rangle \in X^3$ (with x, y variables of sort 0, $\langle x, y\rangle$ a pair term, X^1 a variable of sort 1, and X^3 a variable of sort 3) and $\mathsf{4LQS}^{\mathsf{R}}$ purely universal formulae of the type $(\forall z_1)...(\forall z_n)\varphi_0$ (with z_i variables of sort 0, for $i = 1, \dots, n$, and φ_0 a propositional combination of $\mathsf{4LQS}^{\mathsf{R}}$-quantifier-free atomic formulae) are used to represent the expressive description logic $\mathcal{DL}_{\mathbf{D}}^{4,\times}$, thus yielding a decision procedure for reasoning tasks for $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ such as the consistency of knowledge bases (KBs) and the *Higher Order Conjunctive Query Answering* problem. The latter problem, in particular, includes the most relevant ABox reasoning tasks.

The description logic $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ admits full negation, union and intersection of concepts and abstract roles, concept domain and range, and existential and minimum cardinality restriction on the left-hand side of inclusion axioms. It also

[*] Work supported by *Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM).*

supports role chains on the left-hand side of inclusion axioms and properties on roles such as transitivity, symmetry, reflexivity, and irreflexivity. Thanks to its expressiveness, $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ can represent expressive ontologies, such as, for instance, OntoCeramic [6].

Since existential quantification is admitted only on the left-hand side of inclusion axioms, $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ is less expressive than logics such as $\mathcal{SROIQ}(\mathbf{D})$ [8] as far as the generation of new individuals is concerned. On the other hand, $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ is more liberal than $\mathcal{SROIQ}(\mathbf{D})$ in the definition of role inclusion axioms, since the involved roles are not required to be subject to any ordering relationship, and the notion of simple role is not needed. For example, the role hierarchy presented in [8, page 2] is not expressible in $\mathcal{SROIQ}(\mathbf{D})$, but can be represented in $\mathcal{DL}_{\mathbf{D}}^{4,\times}$. In addition, $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ is a powerful rule language able to express rules with negated atoms such as $Person(?p) \wedge \neg hasHome(?p, ?h) \implies HomelessPerson(?p)$. Notice that rules with negated atoms are not supported by the Semantic Web Rule Language (SWRL).

By resorting to the algorithm introduced in [4], in this paper we present the first effort to implement a KE-tableau based decision procedure for the consistency problem of $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-KBs. Implementation is being carried out in C++, as C++ allows for low level directives and can be easily compiled in several environments. The choice of KE-tableau systems [9] in place of traditional semantic tableaux [11] is motivated by the fact that KE-tableau systems introduce an analytic cut rule allowing the construction of trees whose distinct branches define mutually exclusive situations, thus preventing the proliferation of redundant branches (typical of Smullyan's semantic tableaux). Thus, when a consistent KB is given in input, the procedure yields a KE-tableau whose open branches induce distinct models of the KB. Otherwise, a closed KE-tableau is returned.

Our reasoner, still in beta-testing, is available at the following address:
`https://github.com/dfsantamaria/DL4xD-Reasoner`.

2 Overview of the reasoner

The input of the reasoner is an OWL ontology, serialized in the OWL/XML syntax (see Figure 1). If the ontology meets the $\mathcal{DL}_{\mathbf{D}}^{4,\times}$ requirements, then a parser produces the internal coding of all axioms and assertions of the ontology in set-theoretic terms as a list of strings. Such translation exploits the function θ used in [4] to map $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-KBs to $\mathsf{4LQS^R}$-formulae. Each such string represents either a $\mathsf{4LQS^R}$-quantifier free formula or a $\mathsf{4LQS^R}$ purely universally quantified formula whose quantifiers have been moved as inward as possible. In the subsequent step, the reasoner builds the data-structures required to execute the algorithm, and then it constructs the expansion of each $\mathsf{4LQS^R}$ purely universally quantified formula according to [4, page 9] yielding an expanded (ground) $\mathcal{KB}$, $\Phi_{\mathcal{KB}}$. Subsequently, a KE-tableau $\mathcal{T}_{\mathcal{KB}}$, representing the saturation of $\mathcal{KB}$, is constructed according to Procedure saturate-$\mathcal{DL}_{\mathbf{D}}^{4,\times}$-$\mathcal{KB}$ in [5]. Initially a one-branch KE-tableau $\mathcal{T}_{\mathcal{KB}}$ for $\Phi_{\mathcal{KB}}$ is constructed. Then, $\mathcal{T}_{\mathcal{KB}}$ is expanded till saturation by sistematically applying the E-Rule (elimination rule) and the PB-Rule (principle of bivalence rule) in Figure 2 to formulae of type $\beta_1 \vee \ldots \vee \beta_n$, giving priority to the application of the E-Rule. Once such rules are no longer applicable, for each open branch ϑ of the resulting KE-tableau, literals of type $x = y$ occurring in ϑ are dealt with by storing in ϑ the equivalence class of x and y.

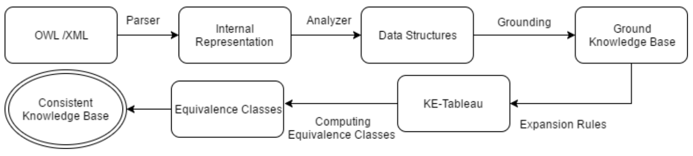

Fig. 1. Execution cycle of the reasoner.

$$\frac{\beta_1 \vee \ldots \vee \beta_n \qquad \mathcal{S}_i^{\overline{\beta}}}{\beta_i} \quad \textbf{E-Rule} \qquad\qquad \frac{}{A \mid \overline{A}} \quad \textbf{PB-Rule}$$

where $\mathcal{S}_i^{\overline{\beta}} := \{\overline{\beta}_1, ..., \overline{\beta}_n\} \setminus \{\overline{\beta}_i\}$, for $i = 1, ..., n$

with A a literal

Fig. 2. Expansion rules for the KE-tableau.

2.1 Some implementation details

We first show how the internal coding of $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-KBs represented in terms of $\mathsf{4LQS^R}$ is defined and how data-structures for the representation of formulae, nodes, and KE-tableaux are implemented. Then we describe the most relevant functions that implement the algorithm.

$\mathsf{4LQS^R}$ elements such as variables, pairs, relators, logic connectors, and quantifiers are internally mapped as strings.

The class `Var` describes $\mathsf{4LQS^R}$ variables by means of a string representing the name, an integer representing the sort of the variable, and an integer indicating whether the variable is bound or free. Purely universally quantified variables and free variables are collected in the vectors `VQL` and `VVL`, respectively, which provide a subvector for each sort of variable.

The operators admitted in $\mathsf{4LQS^R}$ and internally coded as strings are mapped in three distinct vectors that are fields of the class `Operator`.

$\mathsf{4LQS^R}$ atomic formulae are stored using the class `Atom`. The latter has two fields: the integer `atomOp` representing the operator of the formula, and the vector `components`, whose elements point to the variables involved in the atomic formula. $\mathsf{4LQS^R}$ formulae are represented by the class `Formula` having a binary tree-shaped structure, whose nodes contain an object of type `Atom`. The left and the right children contain the left and the right subformula, respectively. The class `Formula` contains a pointer to an object of `Atom`, representing the atomic formula.

The KE-tableau decision procedure exploits the class `Tableau`. This class uses the instances of the class `Node` that represents the nodes of the KE-tableau. `Node` has a tree-shaped structure with four fields: a vector of `Formula` that collects the formulae of the current node, and three pointers to instances of the class `Node` that are the left, the right, and the father node. The set of open branches is collected in the field `openbranches`, whereas the set of closed branches is maintained in the vector `closedbranches`. In addition, the class `Tableau` is provided with a three-dimensional vector of integers `EqSet`, which stores the equivalence classes induced by the atomic formulae of type $X^0 = Y^0$, for each branch ϑ of the tableau and for each variable in ϑ occurring in any atomic formula of type $X^0 = Y^0$.

The task of parsing the ontology from the OWL/XML file is performed by the function `readOWLXML`. The latter takes as input the string obtained by reading the OWL/XML file and returns a vector of strings representing the internal coding of the KB. Once all input formulae have been parsed, the reasoner constructs the expansion of the KB by means of the procedure `expandKB`, which yields the vector of the output formulae (`out`) from the vector of the input formulae (`inpf`). In order to instantiate all the quantified variables, `expandKB` exploits a stack and the vectors `VVL` and `VQL`. After this step, the reasoner checks for atomic clashes in the expanded KB by means of the procedure `checkNodeClash`.

The construction of the KE-tableau is performed by procedure `expandTableau`, which exploits two stacks of type vector of pointers to `Node`. The stack `noncomBranches`, keeps track of the non-complete branches, while `nonfulFormula` keeps track of the non-fulfilled disjunctive formulae. Initially, `expandTableau` attempts to empty the stack `nonfulFormula` by selecting iteratively its elements and applying either the procedure `ERule` or the procedure `PBRule`, respectively implementing the E-Rule and the PB-Rule in Figure 2. The disjuncts of the current formula are stored in a temporary vector and selected iteratively. If a disjunct has its negation on the branch, it is removed from the vector. Once all disjuncts of the formula have been selected, if there is only one element in the stack, then the procedure `ERule` is applied to the disjunctive formula. If the stack contains more than one element, then the procedure `PBRule` is applied. Finally, if the stack is empty, a contradiction is detected, and the branch is added to `closedbranches`. The procedure `expandTableau` terminates when either `noncomBranches` or `nonfulFormula` are empty. When the procedure terminates with some element in `noncomBranches`, such branches are added to the vector `openbranches`. The subsequent phase consists in constructing the set of equivalence classes `EqSet` for each open branch computed by `expandTableau`. `EqSet` is computed by the procedure `computeEqT`. For each open branch in `openbranches`, the procedure searches for formulae of type $X^0 = Y^0$, where X^0 and Y^0 are selected with respect to the ordering provided by the vector `VVL`, and, for each variable, stores the equivalence class in `EqSet`.

The procedure terminates when all open branches of the vector `openbranches` have been analysed. If the vector is not empty, the KB is declared to be consistent.

3 Conclusions

We have presented an ongoing implementation of a KE-tableau based decision procedure for the consistency problem of $\mathcal{DL}_{\mathbf{D}}^{4,\times}$-KBs in terms of set-theoretical $\mathsf{4LQS^R}$-formulae. The reasoner, developed in C++, takes as input OWL ontologies serialized in the OWL/XML format.

Currently, the tool is in its beta-testing phase. We plan to compare it with existing reasoners such as Hermit [7] and Pellet [10], and to provide some benchmarking. Then, we intend to extend the reasoner with the HOCQA procedure [4], thus providing ABox reasoning services. We also plan to allow data type reasoning by integrating Satisfiability Modulo Theories solvers. Moreover, techniques developed in [2, 3] will be used to include reasoning for description logics admitting full existential and universal restrictions. Finally, we intend to implement a parallel version of the software by exploiting Message Passing Interface, since each branch of the KE-tableau can be computed by a single processing unit.

References

1. D. Cantone and M. Nicolosi-Asmundo. On the satisfiability problem for a 4-level quantified syllogistic and some applications to modal logic. *Fundamenta Informaticae*, 124(4):427–448, 2013.
2. D. Cantone, M. Nicolosi-Asmundo, and E. Orłowska. Dual tableau-based decision procedures for some relational logics. In *Proceedings of the 25th Italian Conference on Computational Logic, CEUR-WS Vol. 598, Rende, Italy, July 7-9, 2010*, 2010.
3. D. Cantone, M. Nicolosi-Asmundo, and E. Orłowska. Dual tableau-based decision procedures for relational logics with restricted composition operator. *Journal of Applied Non-Classical Logics*, 21(2):177–200, 2011.
4. D. Cantone, M. Nicolosi-Asmundo, and D. F. Santamaria. A set-theoretic approach to ABox reasoning services. In *Costantini S., Franconi E., Van Woensel W., Kontchakov R., Sadri F., Roman D. Rules and Reasoning. RuleML+RR 2017.*, Lecture Notes in Computer Science, vol 10364. Springer, 2017.
5. D. Cantone, M. Nicolosi-Asmundo, and D. F. Santamaria. A C++ reasoner for the description logic $\mathcal{DL}_{\mathbf{D}}^{4,\times}$. *CoRR*, abs/1707.07545, 2017.
6. D. Cantone, M. Nicolosi-Asmundo, D. F. Santamaria, and F. Trapani. Ontoceramic: an OWL ontology for ceramics classification. In *Proc. of CILC 2015, CEUR-WS, vol. 1459, pp. 122–127*, Genova, July 1-3, 2015.
7. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.
8. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proc. 10th Int. Conf. on Princ. of Knowledge Representation and Reasoning, (Doherty, P. and Mylopoulos, J. and Welty, C. A., eds.)*, pages 57–67. AAAI Press, 2006.
9. M. Mondadori M. D'Agostino. The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation*, 4:285–319, 1994.
10. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.
11. R. M. Smullyan. *First-order Logic.* Dover books on advanced Math. Dover, 1995.