Distributed Resolution for $\mathcal{ALC}$

Anne Schlicht, Heiner Stuckenschmidt

University of Mannheim, Germany
{anne,heiner}@informatik.uni-mannheim.de

Abstract. The use of Description Logic as the basis for Semantic Web Languages has led to new requirements with respect to scalable and non-standard reasoning. In this paper, we address the problem of scalable reasoning by proposing a distributed, complete and terminating algorithm that decides satisfiability of terminologies in $\mathcal{ALC}$. The algorithm is based on recent results on applying resolution to description logics. We show that the resolution procedure proposed by Tammet can be distributed amongst multiple resolution solvers by assigning unique sets of literals to individual solvers. This results provides the basis for a highly scalable reasoning infrastructure for Description logics.

1 Introduction

The use of description logics as one of the primary logical languages for knowledge representation on the Web has created new challenges with respect to reasoning in these logics. In order to be able to support the vision of a semantic web of interrelated ontologies, reasoning procedures have to be highly scalable and able to deal with physically distributed knowledge models. A natural way of addressing these problems is to rely on distributed inference procedures that can distribute the load between different solvers thus reducing potential bottlenecks both in terms of memory and computation time. Currently, almost all the work on description logic reasoning still assumes a centralized approach where the complete terminology has to be present on a single system and all inference steps are are carried out on this system. Exceptions to this rule are approaches like Distributed Description Logics (DDL) [2] that support local reasoning at the price of sacrificing expressiveness in the links between local models and by dropping some formal properties on the level of the overall model. In DDL for example, certain types of inconsistencies are not propagated on the global level. The goal of our work is to support local reasoning in description logics without the need to reduce the expressiveness of links between local models. Based on results in resolution reasoning for description logic [6], we present an algorithm that decides satisfiability of a set of $\mathcal{ALC}$ ontologies. This algorithm allows us to provide distributed reasoning support on sets of terminologies that share non-logical symbols without merging the modules. A possible application is the provision of distributed reasoning support for ($\mathcal{ALC}$ equivalent parts of) OWL ontologies linked by import statements. Our method guarantees that the global semantics is preserved.

The contribution of this paper is twofold: (1) We identify general requirements for a resolution procedure needed to guaranteeing soundness and completeness. (2) we show that the resolution method proposed in [6] satisfies these requirements. The paper is organized as follows: after a brief discussion of related work in the remainder of this section, we first explain the general idea of distributing standard resolution and discuss the problems that have to be solved for guaranteeing completeness. We then discuss ordered resolution as a adequate basis for distribution and its adaptation to $\mathcal{ALC}$ knowledge bases. Finally, we discuss implementation issues of our algorithm as well as possible optimizations.

1.1 Related Work

Modular DL Reasoning Current approaches to distributed reasoning on description logic mostly rely on tableaux methods. Distribution by solving the two choices of nondeterminitic tableaux rules in parallel is difficult as it hampers the application of optimization and blocking strategies. Instead most distributed tableaux approaches try to identify all possible conflicts, i.e. all axioms that might follow from another module and would cause a contradiction and send these as queries to the other modules. So far, this is only done for links with rather restricted expressiveness between the modules.
The most prominent actually distributed T-Box reasoning implementation for ontologies *Distributed Description Logic* (DDL) [2] supports only a special type of links (called bridge rules) between ontologies. The local domains have to be disjoint, i.e. there is no real subsumption between elements of different modules. Like DDL, $\mathcal{E}$*-connections* [3] treat local domains as disjoint and do not support subsumption relations between modules. The same authors are working on a modular ontology framework based on *Conservative Extensions* [5]. This framework, however, does not allow complex interactions between axioms in different modules. This allows the authors to work with local reasoning without any communication between the modules. We think that this approach overshoots the mark and poses too many restrictions on the way axioms might be distributed across different sites.

Distributed Resolution Methods Resolution is used by all successful FOL provers. Approaches to distributed first order reasoning are motivated by efficiency considerations, performance is improved by using multiple processors in parallel. *Roo*[4], for example, is a parallelization of the widely (re)used first order reasoner Otter.
Parallelization differs from the distribution setting, while the former usually uses a shared memory, the latter constitutes a physical separation of modules and resulting higher communication costs. *Partition-Based Reasoning* [1] is a distributed resolution strategy that requires local reasoning to be complete for consequence finding. The requirement inevitably causes derivation of much more clauses than necessary for refutation. Nevertheless the method was shown to speed up some resolution strategies in a parallel setting without communication

costs. The requirement of being complete wrt. consequence finding means that we cannot directly apply this approach to description logics as to the best of our knowledge, there is no method that satisfies this requirement for $\mathcal{ALC}$.

Resolution for Description Logics The main problem with applying FOL approaches for DL is that termination is not guaranteed. Although DL is a decidable fragment of FOL, causing a FOL reasoner to terminate on DL input requires considerable adaption of the algorithm. This paper is based on resolution methods introduced to description logics by [7] that decide satisfiability. The algorithm we present can be seen as a distributed version of the algorithm for deciding $\mathcal{ALC}$ satisfiability described in [6].

2 Distributed Standard Resolution

Analyzing the applicability of the Partition-Based Reasoning approach of [1] to description logic revealed that the only way to avoid redundant derivations is to perform the same tasks as a common resolution reasoner instead of connecting a set of blackbox reasoners. Therefore, the basic idea proposed in this paper is to distribute common standard resolution by allocating each derivation step to a unique site in the distributed system. We assume a DL knowledgebase transformed into first order clauses and give an *allocation function* that maps each occuring clause to the specific module that contains this clause. In general, a complete calculus could require an allocation relation that is not functional but contains pairs $(c, m_i), (c, m_j)$ allocating some clause c to two different modules m_i and m_j. Modules are thus sets of clauses that can be derived from the given knowledge base. Different ontologies that share terms (e.g. one ontologie uses a concept defined in another ontology) can be considered as modules, too. The ontologies can be translated into clauses seperately and an allocation relation can be defined for allocating newly derived clauses to modules.

Definition 1 (Resolution). *For literals $A, \neg B$ clauses $C \vee A, D \vee \neg B$*

$$\textit{Resolution}\ \frac{C \vee A \quad D \vee \neg B}{C\sigma \vee D\sigma}$$

where σ is the most general unifier of A and B.

Resolution is a sound and complete calculus for deciding satisfiability of first order theories. However, additional rules are necessary for obtaining a decision procedure that potentially terminates on realistic input. Most important in practice are *reduction rules*, that delete redundant clauses and thereby reduce the options for new derivations.
In this paper we focus on the distribution of derivation, because reduction can be constricted without sacrificing completeness. We expect that local reduction (i.e. deleting clauses that are redundant wrt. the module they are contained in) is sufficient in practice. Inter-module reduction is discussed briefly in section 5.

Definition 2 (Distributed Resolution). *For a given resolution calculus R and allocation function a, the corresponding distributed calculus is obtained by adding the restriction $\exists m : a(P_1, m), \ldots, a(P_n, m)$ to every rule $\frac{P_1,\ldots,P_n}{D}$ with premises $P_i, i = 1...n$.*

For rules with only one premise this restriction is trivially true and may be omitted. These restrictions constitute a seperation into modules. Implementation is realized by connecting a set of standard reasoners:

- Every module is saturated seperately
- Newly derived clauses are propagated according to the allocation relation

The system stops if the empty clause is derived in one of the modules or all are saturated. In difference to the centralized case, a module that is saturated locally may have to continue reasoning once a new clause is propagated from another module.

2.1 Distributed Resolution Challenge

In general, clauses might have to be allocated to more than one module to make sure that the premises of every derivation step are allocated to the same module. In order to achieve an efficient method, we have to avoid duplication and allocate every clause to as few sites as possible. Even if we assume the case of only two modules where each may use terms defined by the other distributing standard resolution might cause most of the axioms in inferences be copied to the other module. To make sure that every pair of clauses that can be resolved with each other meets in a module, we need to send all clauses containing a foreign term (i.e. a term defined in the other module) to the other module. Unfortunately the resolvent of clauses from different modules is very likely to contain terms from both modules that were only used locally before, thus it has to be copied to both modules. If we extend the approach to multiple modules we can expect further increase of the number of duplicates. Nevertheless, we will show that it is possible to define a distributed resolution method for $\mathcal{ALC}$ that is sound and complete, and terminates. It does not require duplication of axioms, every clause is allocated to exactly one module.

3 Ordered Resolution

The distributed reasoning algorithm we propose relies on a special parameterization of ordered resolution. Intuitively, the idea of ordered resolution is to derive a certain clause only once and not in different ways. If at a point in the derivation process one clause could be resolved on multiple literals with multiple other clauses (one for each literal) it is not necessary to try all permutations of the literal sequence. Instead the literals are ordered and only one sequence of derivations is executed. Moreover some steps can be combined to a single inference (hyperresolution) because the intermediate results are redundant, resulting in

an additional speed up.
Ordered resolution depends on two parameters that can be modified to improve performance e.g. for restricted subsets of first order logic like description logic. First paparmeter is the *order* of literals (for defining the maximal literals) that can be defined on top of an order of predicate symbols, function symbols and constants. The second parameter is the *selection function* that maps every clause C to a subset $S(C)$ of it's negative literals. Essentially the selection function changes the order of literals in a clause, the selected literals are moved to the front. In addition, the combination of multiple inferences for skipping redundant intermediate results (i.e. hyperresolution) is controlled by selection.
We use ordered resolution in combination with positive factoring. The two rules can be combined with hyperresolution into one inference but for the sake of simplicity we use the seperate notation.

Definition 3 (Ordered Resolution).

Ordered resolution:

$$\frac{C \vee A \quad D \vee \neg B}{C\sigma \vee D\sigma}$$

1. σ *is the most general unifier of* A *and* B
2. *either* B *is selected in* $D \vee \neg B$ *or else nothing is selected in* $D \vee \neg B$ *and* $B\sigma$ *is maximal w.r.t.* $D\sigma$
3. $A\sigma$ *is strictly maximal with respect to* $C\sigma$
4. *nothing is selected in* $C\sigma \vee A\sigma$

Positive factoring:

$$\frac{C \vee A \vee B}{C\sigma \vee A\sigma}$$

1. σ *is the most general unifier of* A *and* B
2. $A\sigma$ *is maximal with respect to* $C\sigma \vee B\sigma$
3. *nothing is selected in* $C\sigma \vee A\sigma \vee B\sigma$

For the first rule, $(C \vee A)$ is the side premise and $(D \vee \neg B)$ is main premise. Compared to unrestricted resolution, many derivations are skipped in ordered resolution. A literal A that is not selected in a clause C can only be resolved upon if nothing is selected in the clause and $A\sigma$ is maximal.

3.1 Adaptation to DL

The problem with applying first order methods for DL is loosing decidability. A DL-ontology input to a complete first order reasoner will not terminate in general. [7] showed that using ordered resolution, decidability of $\mathcal{ALC}$ can be preserved by transforming the ontology into a special normal form.

Definition 4 (Description Logic Resolution). *With* R_{DL} *we denote the following reasoning method:*

Clausification *Before skolemization the DL concept definitions are replaced by their* definitorial form[1]*.*

Rules *The resolution rules applied are ordered resolution and positive factoring.*

Ordering *The literal ordering is a lexicographic path ordering (definition below) based on a total precedence $>$ with $f > P > \neg$ for every function symbol f and predicate symbol P. (This implies that $R(x, f(x)) \succ \neg C(x)$ and $D(f(x)) \succ \neg C(x)$, for all function symbols f, and predicates R, C, and D.)*

Selection *The selection function selects every negative binary literal in each clause.*

Definition 5 (Lexicographic Path Ordering).
A lexicographic path ordering (LPO) is a term ordering induced by a well-founded strict precedence $>$ over function, predicate and logical symbols, defined by:
$s = f(s_1, \ldots, s_m) \succ g(t_1, \ldots, t_n) = t$ if and only if
at least one of the following holds

(i) $f > g$ *and* $s \succ t_i$ *for all* i *with* $1 \leq i \leq n$
(ii) $f = g$ *and for some* j *we have* $(s_1, \ldots, s_{j-1}) = (t_1, \ldots, t_{j-!})$, $s_j \succ t_j$ *and* $s \succ t_k$ *for all* k *with* $j < k \leq n$
(iii) $s_j \succeq t$ *for some* j *with* $1 \leq j \leq m$

LPOs have the subterm property, i.e. $E[E'] \succ E'$ for every expression E. Furthermore, if $>$ is total, the LPO induced by $>$ is total on ground terms.

Theorem 1 ($\mathbf{R_{DL}}$ complexity). *For an $\mathcal{ALC}$ knowledge base KB, saturating its definitorial form by R_{DL} decides satisfiability of KB and runs in time exponential in KB.*

For the proof see [7].

4 Distributed Ordered Resolution

Intuitively, the idea for distributing resolution is

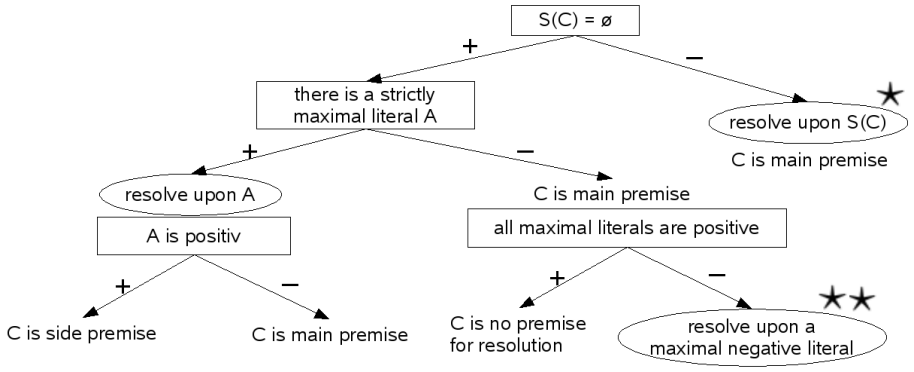
- Every module "hosts" a subset of the literals, i.e. it is responsible for all inferences resolving upon one of these literals.
- Every (derived or stated) clause is moved to modules that host a resolvable literal.

Essential for distributing resolution without duplicating clauses and derivations is to reduce the number of modules that may be responsible for the next derivation to a single module i.e. to define the resolution method in such a way that there is a unique resolvable literal for every clause. Note that literals can be allocated to modules based on their top symbol and a partitioning of symbols.

[1] Due to limited space we refer to [6], chapter 4.3 for definition. Intuitively, the definitorial form can be seen as contrary of unfolding, it splits up nested axioms into simple ones by introducing new concepts.

4.1 Resolvable Literals of a Given Clause

For defining a resolution method that is complete for this communication strategy, we investigate the options for using a given clause in an inference. The first thing we discover from the definition of ordered resolution is that a clause can only be main premise and not side premise if some literal is selected by the selection function. By exploiting the other requirements we obtain the following decision tree (Assuming the decision on the resolvable literal is not affected by unification).



Resolution options for a given clause

If nothing is selected, only a maximal literal can be resolved. If there are multiple maximal literals, then the clause cannot be a side premise. Problematic are multiple negative maximal literals because any of them can be the resolved literal in the next derivation. If all multiple maximal literals are positive only factoring is possible because the clause can neither be a side premise nor a main premise. A given clause can never be side premise and main premise at same time (in different inferences). If the selection function is empty, the first branch on the right is pruned. If the order is total, the branch in the middle is pruned. By exploiting the effects of the selection and ordering on the clause tree, two problematic cases are identified:

- $\star$ Multiple selected literals: If literals from different modules are selected, we cannot allocate the inference to one module.
- $\star\star$ Multiple negative maximal literals, none selected: We have to send the clause to every module that is responsible for derivations resolving upon one of these literals.

Never selecting literals hosted by different modules conflicts with selecting all negative binary literals like required by R_{DL}. Using a total order on literals would avoid the second problem but, even with a total precedence on symbols LPO might be only total on ground literals. Before we address these problems, we make sure that the assumption we made for building the clause tree is true.

4.2 Invariance of Resolvable Literals

The decision tree can only be applied if the resolvable literals for each clause are not changed by unification. The definition of ordered resolution requires each resolved atom $A\sigma$ to be (strictly) maximal with respect to $D\sigma$ ($C\sigma$) but this does in general not imply that A is (strictly) maximal w.r.t. the premise before substitution. For some definitions of selection, unification might also affect the literals to be selected. Fortunately, in the resolution method presented by [6] for $\mathcal{ALC}$, the types of clauses are restricted to five different types and the occuring substitutions are very simple. Furthermore, the selection function is independent of unification. For these clauses depicted below we can show, that $A\sigma$ is (strictly) maximal w.r.t. its clause iff A is (strictly) maximal w.r.t. its clause.

	clause type	type of resolvable literal
1	$R(x, f(x)) \vee \mathbf{P}(x)$	$R(x, f(x))$
2a	$\mathbf{P}(x)$	$(\neg)P(x)$
2b	$\mathbf{P_1}(\mathbf{f}(x)) \vee \mathbf{P_2}(x)$	$(\neg)P(f(x))$
3	$\neg R(x, y) \vee \mathbf{P_1}(x) \vee \mathbf{P_2}(y)$	$\neg R(x, y)$
4	$\mathbf{P}(\mathbf{a})$	$(\neg)P(a)$
5	$(\neg)R(a, b)$	$(\neg)R(a, b)$

In the above table $\mathbf{P}(t)$ (where t is a term) denotes a possibly empty disjunction of the form $(\neg)P_1(t) \ldots (\neg)P_n(t)$. $\mathbf{P}(\mathbf{f}(x))$ denotes a possibly empty disjunction of the form $\mathbf{P_1}(f_1(x)) \ldots \mathbf{P_m}(f_m(x))$. Note that this definition allows each $\mathbf{P_i}(f_i(x))$ to contain positive and negative literals. We choose a slightly different notation of the $\mathcal{ALC}$ clauses with the type 2 clauses subdivided because the resolvable literal is of type $P(f(x))$ if one exists and otherwise $P(x)$.

Analysing all possible types of inferences among these clause will reveal that in all cases the resolvable literals can be determined prior to substitution.
This is obvious for clauses of type 5 (they are unit clauses) and of type 4 because nothing is substituted in these clauses. Furthermore clauses of type 3 contain a selected literal, i.e. the negative binary literal of type $\neg R(x, y)$. For the remaining clauses of type 1 and 2 we show that we can determine the maximal literal of a clause prior to any substitution.

Lemma 1 (Invariance of Maximality). *If a set of $\mathcal{ALC}$ clauses is resolved applying R_{DL}, the maximal literal of a clause of type 1 or 2 is independent of unification.*

Proof. To prove the lemma we apply the definition of the ordering used for R_{DL}, i.e. a LPO (see Definition 5) based on a precedence $>$ of function (f), predicate (P) and logical symbols with ($f > P > \neg$). We demonstrate for any literals A and B of a clause of type 1, 2a or 2b and for all possible substitutions σ that $A\sigma \succ B\sigma$ if and only if $A \succ B$. Possible substitutions are of the form $x \to a$, $x \to y$ and $x \to f(y)$. They do not get more complicated because the definition of the ordering impedes substitution of variables that are arguments

of functions. E.g. $P_1(x)$ will only be unified with $P_2(f(y))$ if no function symbol is contained in the same clause. Without loss of generality we restrict the proof to positive literals, because for every literal A, $\neg A \succ A$ and there is no literal B with $\neg A \succ B \succ A$.

1. The literal $R(x, f(x))\sigma$ is maximal in clauses of type 1 independently of the substitution because $f > P$ and $f(x)\sigma \succ x\sigma$ (subterm property) and by (i) $f(x)\sigma \succ P(x)\sigma$. By (iii) follows $R(x, f(x))\sigma \succ P(x)\sigma$.

2a. $P(x)\sigma \succ Q(x)\sigma$ iff $P > Q$ independent of σ because (ii) and (iii) are not applicable and by the subterm property $P(x)\sigma \succ x\sigma$. Note that the same holds if predicate symbols are replaced by function symbols.

2b. Literals of type $(\neg)P(f(x))$, are larger than any literal of type $P(x)$ independent of substitution by (iii) because $f(x)\sigma \succeq Q(x)\sigma$.
$P(f(x))\sigma \succ Q(g(x))\sigma$ holds iff one of (i),(ii),(iii) hold i.e. $(P > Q \wedge P(f(x))\sigma \succ g(x)\sigma) \vee (P = Q \wedge f(x)\sigma \succ g(x)\sigma) \vee f(x)\sigma \geq Q(g(x))\sigma$. This is equivalent to $(P > Q \wedge f = g) \vee f > g$ independent of substitution, because by (iii) $(P(f(x))\sigma \succ g(x)\sigma$ iff $f(x)\sigma \succeq g(x)\sigma)$ and $(f(x)\sigma \geq Q(g(x))\sigma$ iff $f(x)\sigma \succ g(x)\sigma)$. □

Hence, the resolvable literals of a clause are not affected by any possible substitution. Maximality of a literal in a clause with nothing selected can be computed prior to unification.
Actually the proof of Lemma 1 establishes a stronger result for R_{DL}. The list of clause types on the one hand and the biconditionals $P(x)\sigma \succ Q(x)\sigma$ iff $P > Q$ and $P(f(x))\sigma \succ Q(g(x))\sigma$ iff $(P > Q \wedge f = g) \vee f > g$ on the other hand yield the following corollary.

Corollary 1.
Saturating an $\mathcal{ALC}$ knowledge base by R_{DL}, 1) the selection function never selects more than one literal and 2) the order is total on the literals of a clause that are not selected.

This solves exactly the problems marked by * and ** in the clause decision tree. Thus, there is a unique resolvable literal for every clause c independent of substitution and a unique module $m(c)$ that is responsible for resolving upon this literal.

Theorem 2 (Completeness of Distributed Resolution). *Distributed Ordered Resolution is a complete, terminating procedure for deciding $\mathcal{ALC}$ satisfiability.*

Proof. The theorem follows from the properties of centralized R_{DL} (Theorem 1) and Lemma 1. Factoring is not affected by distribution because it is applied to single clauses. Assume the empty clause is not derived from an unsatisfiable KB by distributed ordered resolution and consider the first inference in the corresponding centralized reasoner that does not occur in the distributed version. The literal that is resolved upon is hosted by a single module. By Lemma 1 this literal is identified to be the unique resolvable literal of each premise when the premise is derived or read. Thus both premises are send to the same module that hosts the literal and the inference is carried out by local resolution. □

5 Optimizations

5.1 Selection Function

The selection function defined for R_{DL} selects at most one literal. A derived clause is very likely to be moved to another module. It would be more efficient to select local literals first. Moreover, if we select multiple literals, we can combine multiple inferences to a so called hyperresolution inference and skip redundant intermediate results. By taking a closer look at the requirements necessary to make sure that the set of $\mathcal{ALC}$ clauses is closed under R_{DL}, we can improve the selection function as follows:

Definition 6 (selection2).

1. *Select nothing from clauses of type 1 and 5*
2. *From type 3 clauses select the negative binary literal.*
3. *From clauses of type 2b select the maximal number of negative literals of type $P(f(x))$ hosted by a single module.*
4. *From clauses of type 2a and 4 select the maximal number of negative literals hosted by a single module.*

We may not select all local literals but selecting all literals hosted by one module has a similar effect. First the clause is moved to that module, but then all local literals are solved in the next inference.

5.2 Reduction

The most important reduction technique is the deletion of clauses that are subsumed by another clause. Deleting newly derived clauses that are subsumed is forward subsumption, backward subsumption refers to the deletion of clauses that are subsumed by a newly derived clause. The naïve approach to distributed subsumption would be to propagate new clauses to all modules that host one of its literals for forward subsumption checks and send it to modules that are responsible for literals equal or smaller than the resolvable literal of the new clause for backward subsumtion.
Fortunately we can again take advantage of the clause typology. Of all different clause types only type 2a can be subsumed by a clause of different type, every other clause can only be subsumed by a clause of the same type. Thus, by indexing the clauses with their types we can skip a large part of the subsumption tests.

6 Summary

We addressed the problem of improving the scalability of description logic reasoning by proposing a distributed resolution method for $\mathcal{ALC}$ terminologies. The algorithm extends ordered resolution with a method for assigning clauses to a unique location at which all possible resolution steps are executed by a local

solver. We analyzed the properties of such a distributed method and identified necessary conditions for guaranteeing completeness and termination of the algorithm. Further, we have shown that the resolution method for $\mathcal{ALC}$ described by Motik satisfies these conditions and can therefore be distributed across different distributed reasoners. Our investigations lay the foundation for the implementation of a large scale reasoning infrastructure for $\mathcal{ALC}$ terminologies and can be seen as a first step towards supporting the vision of the semantic web as a distributed system of interlinked ontologies that can be reasoned upon. It remains to be seen how the proposed algorithm behaves in practice. In particular, we have to find a way to control the cost of communication between local reasoners. The major task in this context is the definition of a suitable distribution strategy based on the nature of the terminologies involved. Further, we have to investigate possible optimizations of the reasoning method some of which are already mentioned in Section 5. Another major issue is the development of distributed reasoning methods for more expressive languages – the final goal is to support reasoning in OWL which is known to be equivalent to $\mathcal{SHOIN}(\mathcal{D})$. A direct extension of the approach proposed here is not possible because dealing with number restrictions requires a different resolution approach, i.e. the use of paramodulation. In this context it is not clear if an assignment of literals to a certain reasoner is still feasible. An investigation of this issue will be the focus of future work.

Acknowledgement

This work was partially supported by the German Science Foundation in the Emmy-Noether Program under contract Stu 266/3-1.

References

1. E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 162(1-2):49–88, 2005.
2. A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.
3. Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Combining owl ontologies using e-connections. *Journal Of Web Semantics*, 4(1), 2005.
4. Ewing L. Lusk, William W. McCune, and John Slaney. Roo: a parallel theorem prover. In *In Proceedings of the 11th CADE*, volume 607 of LNAI, pages 731–734. Springer Verlag, 1992.
5. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence IJCAI-07*. AAAI Press, 2007.
6. Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany, January 2006.
7. Tanel Tammet. *Resolution methods for Decision Problems and Finite Model Building*. PhD thesis, Chalmers University of Technology and University of Göteborg, 1992.