Computational space efficiency and minimal model generation for guarded formulae[⋆]

Lilia Georgieva[1], Ullrich Hustadt[2], Renate A. Schmidt[1]

[1] Department of Computer Science, University of Manchester
Manchester M13 9PL, United Kingdom, {georgiel,schmidt}@cs.man.ac.uk
[2] Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom, U.Hustadt@csc.liv.ac.uk

Abstract. This paper describes a number of hyperresolution-based decision procedures for a subfragment of the guarded fragment. We first present a polynomial space decision procedure of optimal worst-case space and time complexity for the fragment under consideration. We then consider minimal model generation procedures which construct all and only minimal Herbrand models for guarded formulae. These procedures are based on hyperresolution, (complement) splitting and either model constraint propagation or local minimality tests. All the procedures have concrete application domains and are relevant for multi-modal and description logics that can be embedded into the considered fragment.

1 Introduction

The guarded fragment (GF) is a generalisation of the modal fragment of first-order logic. The fragment was introduced in an attempt to explain the good model-theoretic and proof-theoretic properties of modal logics, including the decidability of the satisfiability problem [1]. A variety of decision procedures for the fragment and its extensions have been developed, which utilise different techniques such as ordered resolution, model-theoretic constructions, alternating automata or embedding into second-order logic [7, 9, 12, 13]. However, the devised decision procedures have some drawbacks. In particular, they exhibit at least double exponential worst-case time and space complexity [7, 12] which is in contrast to the low complexity of the satisfiability problem of basic modal logic. Moreover, extensions of the guarded fragment with transitivity or number restrictions lead to undecidability [10], even though modal logics extended with transitivity or number restrictions are decidable. This shows that the guarded fragment as a whole is too general and expressive and cannot thoroughly explain the good computational properties of modal logics and related description logics. A natural question arises whether there are more restricted, but yet expressive fragments which provide more suitable logics and for which there are

[⋆] The authors thank Peter Baumgartner for drawing our attention to Niemelä's work and raising the question of the computational complexity of minimal model generation. The work of the first and third authors is supported by EPSRC Research Grant GR/M36700.

algorithms with better worst-case complexity and that possibly employ inference techniques similar to those prominently used for modal and description logics. Such a subfragment, called GF1$^-$, is identified by Lutz, Sattler and Tobies [18], who describe a tableaux procedure which decides GF1$^-$ and prove that under certain assumptions the satisfiability problem of GF1$^-$ is PSPACE-complete.

Our investigations are situated in the framework of resolution which is well-studied and well-mechanised. In previous work [11] we describe a hyperresolution decision procedure for GF1$^-$ and certain generalisations of GF1$^-$, which also include formulae outside the guarded and loosely guarded fragment. This procedure polynomially simulates the tableaux procedure for GF1$^-$ of [18] and forms a generalisation of the selection-based resolution procedure which polynomially simulates tableaux procedures for modal logics and description logics [8, 15]. However, in the worst case the procedure has double exponential space complexity.

The first part of this paper describes a resolution decision procedure for GF1$^-$ with optimal space complexity (Section 3). This procedure is based on the hyperresolution decision procedure of [11] and uses the so-called trace technique [21] which is utilised in PSPACE tableaux procedures for description (and modal) logics. The presentation of the algorithm is sufficiently detailed and preserves the essential structure of the main inference loop found in state-of-the-art theorem provers such as Gandalf, OTTER, SPASS and Vampire, and is thus easily implementable.

The second part of the paper considers the problem of generating minimal Herbrand models for GF1$^-$ (Section 4). The generation of minimal (Herbrand) models has been shown to be useful in a number of applications [2, 4] and we believe that modal logics and generalisations like GF1$^-$ could provide expressive languages for the specification of related applications in the area of multi-agent systems [19].

There are various approaches to generating minimal Herbrand models with hyperresolution [5, 6, 14, 20] which, with the exception of [6], have been applied only to propositional clause sets. We focus on two of these approaches. The first is based on an extension of our resolution-based decision procedure for GF1$^-$ [11] by a model constraint propagation rule which ensures that only minimal models are generated. This generalises the approach and results of Bry and Yahya [6]. The second approach avoids the need for model constraints by using a variant of a local minimality test proposed by Niemelä [20]. Unfortunately, Niemelä's approach requires that the complement of a Herbrand model is finite, which is not the case for GF1$^-$. We show how this problem can be solved, and discuss and compare the space and time complexity of procedures based on these two approaches to minimal model generation for GF1$^-$.

Both minimal Herbrand model generation approaches are applicable to the clausal class $\mathcal{PVD}$ [16] and to all modal and description logics that can be embedded into GF1$^-$, the description logic $\mathcal{ALC}$, and the modal logic $K_{(m)}(\cap, \cup, \smile)$, which is defined over families of binary relations closed under intersection, union and converse [8].

2 Preliminaries

The notational convention is as follows. We denote first-order variables by x, y, z, terms by s, t, u, constants by a, b, functions by f, g, h, predicate symbols by P, Q, G, atoms by A, A_1, A_2, literals by L, clauses by C, formulae by φ, ϕ, ψ, ϑ, and sets of clauses by N. An over-line indicates a sequence. If $\overline{s} = (s_1, \ldots, s_n)$ then $\overline{f(\overline{s})}$ denotes a sequence of terms of the form $f_k(s_1, \ldots, s_n)$.

Let each predicate symbol G be associated with a unique grouping (i, j) where i, $j > 0$. Then: (i) If φ is an atomic formula, φ is in GF1$^-$. (ii) If φ is in GF1$^-$ and G has grouping (i, j), then $\exists\overline{y}\,(G(\overline{x}, \overline{y}) \wedge \varphi(\overline{y}))$, $\forall\overline{y}\,(G(\overline{x}, \overline{y}) \rightarrow \varphi(\overline{y}))$, $\exists\overline{x}\,(G(\overline{x}, \overline{y}) \wedge \varphi(\overline{x}))$, and $\forall\overline{x}\,(G(\overline{x}, \overline{y}) \rightarrow \varphi(\overline{x}))$ are GF1$^-$ formulae provided $\overline{x}$ is an sequence of variables of length i, $\overline{y}$ is a sequence of variables of length j, and $\overline{x} \cap \overline{y} = \emptyset$. Repetitions and permutations of variables are allowed. The atoms $G(\overline{x}, \overline{y})$ are called *guards.* (iii) $\top$ and $\bot$ are in GF1$^-$. And, (iv) GF1$^-$ is closed under Boolean connectives. If $\overline{s}$ and $\overline{t}$ are arbitrary sequences of terms of length i and j, then $\mathcal{G}(\overline{s}, \overline{t})$ represents both $G(\overline{s}, \overline{t})$ and $G(\overline{t}, \overline{s})$.

The procedures described in this paper are based on the resolution decision procedure for GF1$^-$ presented in [11]. It proceeds as follows. First a given GF1$^-$ formula is transformed (by a polynomial time algorithm) into a set of clauses, which have the following forms.

$$\begin{array}{ll}
Q_\varphi(\overline{a}) & \\
\neg Q_\varphi(\overline{x}) \vee \neg P(\overline{x}) & \text{if } \varphi = \neg P(\overline{x}) \\
\neg Q_\varphi(\overline{x}) \vee \neg \mathcal{G}(\overline{x}, \overline{y}) \vee Q_\psi(\overline{y}) & \text{if } \varphi = \forall\overline{y}\,(\mathcal{G}(\overline{x}, \overline{y}) \rightarrow \psi(\overline{y})) \\
\neg Q_\varphi(\overline{x}) \vee \mathcal{G}(\overline{x}, \overline{f(\overline{x})}) & \\
\neg Q_\varphi(\overline{x}) \vee Q_\psi(\overline{f(\overline{x})}) & \text{if } \varphi = \exists\overline{y}\,(\mathcal{G}(\overline{x}, \overline{y}) \wedge \psi(\overline{y})) \\
\neg Q_\varphi(\overline{z}) \vee Q_\psi(\overline{x}) & \\
\neg Q_\varphi(\overline{z}) \vee Q_\phi(\overline{y}) & \text{if } \varphi = \psi(\overline{x}) \wedge \phi(\overline{y}),\ \overline{z} = \overline{x} \cup \overline{y} \\
\neg Q_\varphi(\overline{z}) \vee Q_\psi(\overline{x}) \vee Q_\phi(\overline{y}) & \text{if } \varphi = \psi(\overline{x}) \vee \phi(\overline{y}),\ \overline{z} = \overline{x} \cup \overline{y}
\end{array}$$

Here, $\overline{f(\overline{x})}$ is a sequence of (distinct) Skolem terms introduced for the sequence of existentially quantified variable $\overline{y}$ in $\exists\overline{y}\,(\mathcal{G}(\overline{x}, \overline{y}) \wedge \psi(\overline{y}))$.

Second, a hyperresolution calculus is used to determine satisfiability of the clause set. The calculus, denoted by $\mathsf{R}^{\mathsf{hyp}}$, uses the following expansion rules:

Deduce: $$\frac{N}{N \cup \{C\}}$$
where C is a resolvent or a factor.

Splitting: $$\frac{N \cup \{C_1 \vee C_2\}}{N \cup \{C_1\} \quad | \quad N \cup \{C_2\}}$$
where C_1 and C_2 are variable disjoint.

The resolution and factoring inference rules are:

Hyperresolution: $$\frac{C_1 \vee A_1 \quad \ldots \quad C_n \vee A_n \quad \neg A_{n+1} \vee \ldots \vee \neg A_{2n} \vee D}{(C_1 \vee \ldots \vee C_n \vee D)\sigma}$$
where (i) σ is the most general unifier such that $A_i\sigma = A_{n+i}\sigma$ for every i, $1 \leq i \leq n$, and (ii) $C_i \vee A_i$ and D are positive clauses, for every i, $1 \leq i \leq n$. The right most premise in the rule is referred to as the *negative* premise and all other premises are referred to as *positive* premises.

Factoring: $$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$
where σ is the most general unifier of A_1 and A_2.

A *derivation* in $\mathsf{R}^{\mathsf{hyp}}$ from a set of clauses N is a finitely branching, ordered tree T with root N and nodes are sets of clauses. The tree is constructed by applications of the expansion rules to the leaves. We assume that no hyperresolution or factoring inference is computed twice on the same branch of the derivation. Any path $N(= N_0), N_1, \ldots$ in a derivation T is called a *closed branch* in T iff the clause set $\bigcup_j N_j$ contains the empty clause, otherwise it is called an *open branch*. We call a branch B in a derivation tree *complete* (with respect to $\mathsf{R}^{\mathsf{hyp}}$) iff no new successor nodes can be added to the endpoint of B by $\mathsf{R}^{\mathsf{hyp}}$, otherwise it is called an *incomplete branch*. A derivation is *complete* iff all of its branches are either closed or complete. A derivation T is a *refutation* iff every path $N(= N_0), N_1, \ldots$ in it is a closed branch, otherwise it is called an *open derivation*. A *branch selection function* is a function, mapping an open derivation tree to one of its open branches. A derivation T from N is called *fair* if for any path $N(= N_0), N_1, \ldots$ in T, with *limit* $N_\infty = \bigcup_j \bigcap_{k \geq j} N_k$, it is the case that each clause C that can be deduced from non-redundant premises in N_∞ is contained in some N_j. Note that for a finite path $N(= N_0), N_1, \ldots N_n$, the limit N_∞ is equal to N_n.

Theorem 1 ([3]). *Let T be a fair derivation from a set N of clauses. Then: (i) If $N(= N_0), N_1, \ldots$ is a path with limit N_∞, N_∞ is saturated up to redundancy. (ii) N is satisfiable if and only if there exists a path in T with limit N_∞ such that N_∞ is satisfiable. (iii) N is unsatisfiable if and only if for every path $N(= N_0), N_1, \ldots$ the clause set $\bigcup_j N_j$ contains the empty clause.*

We restrict our attention to derivations generated by strategies such that the positive premises of any hyperresolution step are positive ground unit clauses. For GF1$^-$ this can be achieved by performing suitable splitting and factoring inferences before hyperresolution inferences. Since we are able to prove termination of any such derivation for the clausal set rendered by formulae in GF1$^-$, any such strategy is fair.

Theorem 2 ([11]). *Let φ be a GF1$^-$ formula and let N be the corresponding clause set. Then: (i) Any $\mathsf{R}^{\mathsf{hyp}}$ derivation from N terminates. (ii) φ is unsatisfiable iff all branches in any complete $\mathsf{R}^{\mathsf{hyp}}$ derivation with root N are closed.*

We now recall the definitions of some notions from [11] (which are closely related to notions introduced in [18]). By the *class of GF1$^-$ clause sets* we mean the class of all clause sets N for which a GF1$^-$ formula φ exists such that N is the clausal form of φ as described above. Let N be a GF1$^-$ clause set. A function symbol f_k is said to be *associated with* a predicate symbol Q iff N contains a definitional clause of the form $\neg Q(\overline{x}) \vee \mathcal{G}(\overline{x}, \overline{f(\overline{x})})$ in which f_k occurs. A set $\{t_1, \ldots, t_n\}$ (or sequence $\overline{t} = (t_1, \ldots, t_n)$) of ground terms is called a *uni-node* iff either each t_i, $1 \leq i \leq n$, is a constant, or there exists a predicate symbol Q and a sequence of ground terms $\overline{s}$, such that each t_i, $1 \leq i \leq n$, has the form $f_k(\overline{s})$, where f_k is a function symbol associated with Q. A uni-node X_2 is called a *direct successor* of a uni-node X_1 iff there is a predicate symbol Q such that for each element t of X_2 there is a function symbol f_k, associated with Q, and $t = f_k(\overline{s})$, where $\overline{s}$ is a sequence of precisely the elements of X_1. A set (or sequence) of ground terms is called a *bi-node* iff it can be presented as a union $X_1 \cup X_2$ of two non-empty disjoint uni-nodes X_1 and X_2 such that X_2 is a direct successor of X_1. A ground literal is a *uni-node* (*bi-node*) iff the set of its arguments is a uni-node (bi-node). A clause is a *uni-node* (*bi-node*) iff the set of the arguments of all literals in it is a uni-node (bi-node). The *successor relation* on uni-nodes is the transitive closure of the direct successor relation. (See [11] for examples.)

3 A space efficient resolution decision procedure

Due to space restrictions we only describe the modifications necessary to turn the main procedure of a standard saturation based theorem prover with splitting [22] into a space efficient decision procedure for GF1$^-$ and stipulate the main results.

The procedure exploits the tree structure of uni-nodes induced by bi-nodes of the form $\mathcal{G}(\overline{s}, \overline{t})$. With each uni-node $\overline{s}$ we can associate all unit clauses of the form $Q(\overline{s})$. Then, inferences never involve premises $Q(\overline{s})$ and $Q'(\overline{t})$ associated with distinct uni-nodes $\overline{s}$ and $\overline{t}$. Thus, the sets of clauses associated with different uni-nodes can be investigated independently of each other in a depth-first manner. This is sometimes called the *trace technique*. However, similar as for the modal logic KB or description logics with inverse roles, clauses associated with a uni-node $\overline{t}$ can be used to derive additional clauses associated with a uni-node $\overline{s}$ such that $\overline{t}$ is a direct successor of $\overline{s}$. This suggests a way of investigating the uni-nodes of the tree structure that minimises the space required to store the uni-node clauses associated with the nodes and goes as follows.

Suppose we are currently investigating a uni-node $\overline{s}$. We first try to derive all uni-node clauses associated with $\overline{s}$. If one or more of these clauses is a non-unit clause, we apply the splitting rule which generates additional branches in the derivation tree. If we derive a contradiction, then the current branch is closed and we backtrack to an alternative open branch. If no open branch exists, then the clause set and also the GF1$^-$ formula under consideration are unsatisfiable. If we do not derive a contradiction, then we continue by deriving all bi-nodes of the form $\mathcal{G}(\overline{s}, \overline{f(\overline{s})})$ providing us with the information of which direct successors of $\overline{s}$ exist. We continue by investigating each of these successor nodes independently. Using the clauses stemming from the existentially and universally quantified formulae of the GF1$^-$ formula under consideration we first establish an initial set of clauses associated with a particular successor node $\overline{f(\overline{s})}$. We then recursively call the main procedure for this initial set of clauses. The recursive call can lead to three different results. First, we may derive a contradiction by an application of positive hyperresolution to the clauses associated with one of the uni-nodes. Again, the current branch of the derivation is closed and we move to an alternative open branch of the derivation (no investigation of other successor nodes of $\overline{s}$ is necessary). Second, we may not derive a contradiction, but while considering

Procedure 1 Space efficient resolution decision procedure

```
Procedure ResolutionProver($\overline{s}$, $\mathcal{US}$, $\mathcal{WO}$)
local $N$, $\mathcal{NEW}$, $\mathcal{R}$, Given, Flag, $\overline{t}$;
begin
  while ($\mathcal{US} \neq \emptyset$ and ($\bot \notin \mathcal{US}$ or not StackEmpty(Stack))) do
    if ($\bot \in \mathcal{US}$) then
      (Stack, $\mathcal{US}$, $\mathcal{WO}$) := backtrack(Stack, $\mathcal{US}$, $\mathcal{WO}$)
    else
      ($N$, Flag, $\mathcal{US}$) := choose($\mathcal{US}$, $\mathcal{WO}$);
      if (Flag $\in$ {BOOLEAN, DEFAULT, UNI-NODE, UNIV}) then
        (Given, $N$) := PickAndDelete($N$);
        if (Splittable(Given)) then
          $\mathcal{NEW}$ := FirstSplitCase(Given);
          Stack := push(Stack, SecondSplitCase(Given))
        else
          $\mathcal{WO}$ := $\mathcal{WO} \cup$ Given;
          $\mathcal{NEW}$ := inf(Given, $\mathcal{WO}$);
        ($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$) := ired($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$);
        $\mathcal{US}$ := $\mathcal{US} \cup \mathcal{NEW}$
      else (* Flag $\in$ {EXIST, BI-NODE} *)
        ($\mathcal{R}$, $\mathcal{US}$, $\mathcal{WO}$) := InvestigateAllSuccessors($\overline{s}$, $N$, Flag, $\mathcal{US}$, $\mathcal{WO}$);
        if (restart($\overline{t}$) $\in \mathcal{R}$) then return($\mathcal{R}$)
  return($\mathcal{US}$)
end
```

$\overline{f(\overline{s})}$ we derive some additional clause associated with $\overline{s}$. In this case we delete the clauses associated with the successor node $\overline{f(\overline{s})}$, backtrack to the node $\overline{s}$, add the newly derived clause, and restart the investigation of this node. The node $\overline{f(\overline{s})}$ will then be revisited later. This has been referred to as the *reset-restart* technique [18]. Third, we may neither derive a contradiction nor additional information about a predecessor node. Then we can delete all clauses associated with $\overline{f(\overline{s})}$ and turn to some other successor node $\overline{g(\overline{s})}$. If there is no other successor node, then the clause set is satisfiable.

The main procedure ResolutionProver presented in Procedure 1 follows the search strategy just outlined. ResolutionProver operates on two sets of clauses $\mathcal{US}$ and $\mathcal{WO}$ (the set of *usable clauses* and the set of *worked-off clauses*). The set $\mathcal{WO}$ contains all the clauses that have already been used as (positive) premises in inference steps (or can never be used as positive premises) and the set $\mathcal{US}$ contains all the clauses that still need to be considered as (positive) premises. Let N denote the clauses obtained from a given GF1$^-$ formula φ. Initially $\mathcal{US}$ is the singleton set $\{Q_\varphi(\overline{a})\}$ while $\mathcal{WO}$ is $N - \{Q_\varphi(\overline{a})\}$.

The procedure proceeds in a while-loop which terminates if either the set $\mathcal{US}$ is empty or $\mathcal{US}$ contains the empty clause and there are no more alternative open branches in the derivation tree generated by applications of the splitting rule that can be considered. In the while-loop we choose some of the clauses from $\mathcal{US}$, perform the inferences possible with these clauses and update $\mathcal{US}$ and $\mathcal{WO}$ accordingly. Note that ResolutionProver takes as an additional argument a

Procedure 2 The procedure choose for guarded formulae

```
Procedure choose($\mathcal{US}$, $\mathcal{WO}$)
local $N$, $\mathcal{G}$, $Q$, $Q'$, $C$, $\overline{t}$, $\overline{x}$, $\overline{f(\overline{x})}$;
begin
  if ($Q(\overline{t}) \in \mathcal{US}$ for some $Q$, not newly introduced, and $Q(\overline{t})$ is a uni-node) then
    return($\{Q(\overline{t})\}$, UNI-NODE, $\mathcal{US} - \{Q(\overline{t})\}$)
  else if ($Q(\overline{t}) \in \mathcal{US}$ where $\neg Q(\overline{x}) \vee \neg\mathcal{G}(\overline{x}, \overline{y}) \vee Q'(\overline{y}) \in \mathcal{WO}$) then
    return($\{Q(\overline{t})\}$, UNIV, $\mathcal{US} - \{Q(\overline{t})\}$)
  else if ($Q(\overline{t}) \in \mathcal{US}$ where $\neg Q(\overline{x}) \vee C$ is a Boolean definitional clause in $\mathcal{WO}$) then
    return($\{Q(\overline{t})\}$, BOOLEAN, $\mathcal{US} - \{Q(\overline{t})\}$)
  else if ($C \in \mathcal{WO}$ where $C$ is ground, but non-unit) then
    return($\{C\}$, BOOLEAN, $\mathcal{US} - \{C\}$)
  else if ($Q(\overline{t}) \in \mathcal{US}$ where $\neg Q(\overline{x}) \vee \mathcal{G}(\overline{x}, \overline{f(\overline{x})}) \in \mathcal{WO}$) then
    $N := \{Q(\overline{t}) \in \mathcal{US} \mid \neg Q(\overline{x}) \vee \mathcal{G}(\overline{x}, \overline{f(\overline{x})}) \in \mathcal{WO}$ or $\neg Q(\overline{x}) \vee Q'(\overline{f(\overline{x})}) \in \mathcal{WO}\}$;
    return($N$, EXIST, $\mathcal{US} - N$)
  else if ($\mathcal{G}(\overline{s}, \overline{t}) \in \mathcal{US}$ for some $\mathcal{G}$, not newly introduced, and $\mathcal{G}(\overline{s}, \overline{t})$ is a bi-node)
  then
    return($\{\mathcal{G}(\overline{s}, \overline{t})\}$, BI-NODE, $\mathcal{US}$)
  else if ($C$ is some (arbitrary) clause in $\mathcal{WO}$) then
    return($\{C\}$, DEFAULT, $\mathcal{US} - \{C\}$)
end
```

sequence of terms $\overline{t}$ which is the uni-node the procedure is currently working on. The uni-node the procedure ResolutionProver is initially working on will be $\overline{a}$.

The procedure choose selects the clauses which will be the next to serve as one of the premises of $\mathsf{R}^{\mathsf{hyp}}$, it is a so-called *clause selection function.* Normally, it selects one clause according to some heuristic taking the 'complexity' and 'age' of clauses into account. Instead the modified version of choose presented in Procedure 2 chooses a set of clauses in a way that allows us to take advantage of the trace technique. For this purpose we have to delay the consideration of clauses related to existentially quantified formulae until all other clauses have been dealt with. To ensure this, choose not only selects potential positive premises accordingly, but also passes information about the corresponding negative premises back to the main procedure in the form of a flag. In all cases, except if the value of the flag is 'BI-NODE', choose also returns the set $\mathcal{US}$ from which the chosen clauses have been removed.

Three additional procedures used by ResolutionProver are PickAndDelete, inf, and ired. Given a set N of clauses the procedure PickAndDelete selects a clause C from a set N of clauses according to some appropriate heuristic and returns it together with $N - \{C\}$. The actual inferences by hyperresolution and factoring[1] are performed by the procedure inf. Tautologies and subsumed clauses are removed from the sets $\mathcal{NEW}$, $\mathcal{WO}$, and $\mathcal{US}$ by the procedure ired.

If choose returns the flag value 'EXIST' or 'BI-NODE', then the investigation of the current uni-node is complete and we are about to turn to its successor

[1] For completeness we do not need factoring for GF1$^-$ clause sets because all derived non-unit clauses are ground and can be split.

Procedure 3 Investigation of all successor nodes

```
Procedure InvestigateAllSuccessors($\overline{t}$, $N$, Flag, $\mathcal{US}$, $\mathcal{WO}$)
local $\mathcal{AUXUS}$, $\mathcal{AUXWO}$, $\mathcal{B}$, $\mathcal{G}$, $\mathcal{NEW}$, $\mathcal{R}$, $\mathcal{UNIVS}$, Given, $Q$, $Q'$, $\overline{s}$;
begin
  $\mathcal{AUXUS} := \mathcal{US} \cup N$;
  $\mathcal{AUXWO} := \mathcal{WO}$;
  if (Flag $\in$ EXIST) then
    while ($N \neq \emptyset$) do
      (Given, $N$) := PickAndDelete($N$);
      $\mathcal{WO} := \mathcal{WO} \cup \{\text{Given}\}$;
      $\mathcal{NEW}$ := inf(Given, $\mathcal{WO}$);
    ($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$) := ired($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$);
    $\mathcal{US} := \mathcal{US} \cup \mathcal{NEW}$;
  $\mathcal{UNIVS} := \{Q(\overline{t}) \in \mathcal{US} \cup \mathcal{WO} \mid \neg Q(\overline{x}) \vee \neg \mathcal{G}(\overline{x}, \overline{y}) \vee Q'(\overline{y}) \in \mathcal{WO}\}$;
  $\mathcal{B} := \{\mathcal{G}(\overline{t}, \overline{s}) \mid \mathcal{G}(\overline{t}, \overline{s}) \in \mathcal{US} \cup \mathcal{WO}$ and $\mathcal{G}(\overline{t}, \overline{s})$ is a bi-node$\}$;
  while ($\mathcal{B} \neq \emptyset$ and $\perp \notin \mathcal{US}$) do
    (Given, $\mathcal{B}$) := PickAndDelete($\mathcal{B}$);
    ($\mathcal{R}$,$\mathcal{US}$) := InvestigateOneSuccessor($\overline{t}$, $\mathcal{US}$, $\mathcal{WO}$, $\mathcal{UNIVS}$, Given);
    if (restart($\overline{s}$) $\in \mathcal{R}$) then
      if (depth($\overline{s}$) = depth($\overline{t}$)) then
        $\mathcal{US} := \mathcal{AUXUS} \cup \mathcal{R} - \{\text{restart}(\overline{s})\}$;
        $\mathcal{WO} := \mathcal{AUXWO}$;
        $\mathcal{B} := \emptyset$;
        $\mathcal{R} := \emptyset$;
      else
        $\mathcal{B} := \emptyset$;
  return($\mathcal{R}$, $\mathcal{US}$, $\mathcal{WO}$)
end
```

nodes, which is done by the procedure InvestigateAllSuccessors in Procedure 3. To do this, we first have to establish which successor nodes exist for the current uni-node. If the value of the flag is 'EXIST', then the set of selected clauses returned by choose contains all the unit clauses which can be resolved with clauses stemming from existentially quantified subformulae of φ to generate clauses of the form $\mathcal{G}(\overline{t}, \overline{s})$ and $Q_\psi(\overline{s})$ where $\overline{s}$ is a direct successor of $\overline{t}$. These clauses are computed and added to $\mathcal{US}$. Besides clauses of the form $Q_\psi(\overline{s})$ which already provide information about the successor node $\overline{s}$, additional information can be derived using unit clauses $Q_\vartheta(\overline{t})$ together with $\mathcal{G}(\overline{t}, \overline{s})$ and $\neg Q_\vartheta(\overline{x}) \vee \neg \mathcal{G}(\overline{x}, \overline{y}) \vee Q_\phi(\overline{y})$. We compute the set of all these unit clauses and assign it to $\mathcal{UNIVS}$ and we also compute the set of all bi-nodes of the form $\mathcal{G}(\overline{t}, \overline{s})$ and assign it to $\mathcal{B}$. As described in the outline of the search strategy of the main procedure, we want to investigate the successor nodes independently of each other. Therefore, we consider each element of $\mathcal{B}$ using the procedure InvestigateOneSuccessor presented in Procedure 4. InvestigateOneSuccessor adds the element of $\mathcal{B}$ to the set $\mathcal{WO}$ to form the set $\mathcal{WO}'$ and uses it and the clauses in $\mathcal{UNIVS}$ to compute additional uni-node clauses associated with the successor node it investigates and stores them in $\mathcal{NEW}$. If we already derive the empty clause at this point, then

Procedure 4 Investigation of a single successor node

```
Procedure InvestigateOneSuccessor($\overline{t}$, $\mathcal{US}$, $\mathcal{WO}$, $\mathcal{UNIVS}$, Binode)
local $\mathcal{WO}'$, $\mathcal{NEW}$, $\mathcal{R}$, Given, $\overline{s}$;
begin
  $\mathcal{WO}' := \mathcal{WO} \cup \{\text{Binode}\}$;
  $\mathcal{NEW} := \emptyset$;
  while ($\mathcal{UNIVS} \neq \emptyset$ and $\bot \notin \mathcal{US}$) do
    (Given, $\mathcal{UNIVS}$) := PickAndDelete($\mathcal{UNIVS}$);
    $\mathcal{NEW}$ := inf(Given, $\mathcal{WO}$);
    ($\mathcal{NEW}$, $\mathcal{WO}'$, $\mathcal{US}$) := ired($\mathcal{NEW}$, $\mathcal{WO}'$, $\mathcal{US}$);
  if ($\bot \in \mathcal{NEW}$) then
    return($\{\bot\}$, $\mathcal{US} \cup \mathcal{NEW}$)
  else if ($Q_\vartheta(\overline{s}) \in \mathcal{NEW}$ such that $\overline{s}$ is not a successor of $\overline{t}$) then
    return($\{Q_\psi(\overline{s}) \mid Q_\psi(\overline{s}) \in \mathcal{NEW}\} \cup \{\text{restart}(\overline{s})\}$, $\mathcal{US}$)
  else
    $\overline{s}$ := SuccessorOf($\overline{t}$, Binode);
    $\mathcal{R} := \{Q(\overline{s}) \mid Q(\overline{s}) \in \mathcal{US}\}$;
    $\mathcal{NEW} := \mathcal{NEW} \cup \mathcal{R}$;
    $\mathcal{US} := \mathcal{US} - \mathcal{R}$;
    $\mathcal{R}$ := ResolutionProver($\overline{s}$, $\mathcal{NEW}$, $\mathcal{WO}'$);
    if (restart($\overline{s}$) $\in \mathcal{R}$) then return($\mathcal{R}$, $\mathcal{US}$) else return($\emptyset$, $\mathcal{US} - \{\text{Binode}\}$)
end
```

InvestigateOneSuccessor returns the set $\mathcal{US}$ containing the empty clause to the main procedure which will conclude that the clause set under consideration is unsatisfiable. If we do not derive a contradiction, but additional information about a node $\overline{s}$ which is not a successor of $\overline{t}$, the uni-node under investigation by the main procedure, then we return the additional information to InvestigateAllSuccessors together with an instruction to restart the investigation for $\overline{s}$. This instruction is encoded in a special unit clause restart($\overline{s}$) that we add to the result returned. Otherwise, we determine the successor node $\overline{s}$ of $\overline{t}$ by using the function SuccessorOf, collect the information about $\overline{s}$ contained in $\mathcal{US}$ in $\mathcal{R}$, add the clauses in $\mathcal{R}$ to $\mathcal{NEW}$ and delete them from $\mathcal{US}$, and call the main procedure ResolutionProver with parameters $\overline{s}$, $\mathcal{NEW}$, and $\mathcal{WO}'$. The important point here is that the set of usable clauses on which ResolutionProver will work only contains uni-nodes associated with $\overline{s}$.

Theorem 3. *The following holds for the refined decision procedure: (i) The procedure* choose *ensures a fair selection of positive premises. (ii) The refined decision procedure is sound. (iii) The refined decision procedure is complete.*

In general the space requirement of the refined decision procedure is still exponential in the size of the given GF1$^-$ formula. To achieve the polynomial space bound we have to assume that either (i) the predicates have bounded arity, or (ii) that each subformula of a GF1$^-$ formula has a bounded number of free variables. These assumptions have been made in [12] in order to establish that the satisfiability problem of the guarded fragment is complete for exponential time and are weaker than the ones made in [18].

Theorem 4. *Under one or both of the assumptions (i) and (ii) the refined procedure decides the satisfiability of GF1$^-$ formulae in polynomial space.*

There are three important points worth noting. (i) Even with either, or both, of the assumptions made above a standard saturation theorem prover based on $\mathsf{R}^{\mathsf{hyp}}$ would require exponential space, that is, the modifications described in this section are essential for Theorem 4. (ii) Since the procedure deletes clauses which are not redundant by the standard definition, the set of positive ground unit clauses it keeps, does not necessarily form a Herbrand model of the clause set N at any point in the derivation. (iii) While our procedure imposes a certain order on the selection of clauses, this order still provides flexibility for further refinement and heuristics. In particular, all the standard heuristics used in tableaux decision procedures for description logics can still be utilised. On the other hand most of the heuristics found in resolution theorem provers will provide little guidance due to the particular normal form used.

4 Generating minimal Herbrand models

A *Herbrand interpretation* is a set of ground atoms. By definition a ground atom A is *true* in the interpretation H if $A \in H$ and it is *false* in H if $A \notin H$, $\top$ is true in all interpretations and $\bot$ is false in all interpretations. A literal $\neg A$ is true in H iff A is false in H. A clause C is true in an interpretation H iff for all ground substitutions σ there is a literal L in $C\sigma$ which is true in H. A set N of clauses is true in H iff all clauses in N are true in H. If a set N of clauses is true in an interpretation H then H is referred to as a *Herbrand model* of N. An interpretation H is a *minimal Herbrand model* for a set N of clauses iff H is a Herbrand model of N and for no Herbrand model H' of N, $H' \subset H$ holds.

A clause C is *range restricted* iff the set of variables in the positive part of C is a subset of the variables of the negative part of C. This means that a positive clause is range restricted only if it is a ground clause. A clause set is range restricted iff it contains only range restricted clauses.

Lemma 1. *Let N be the clausal form of a GF1$^-$ formula φ. Then, any clause in N and any clause derived from N by $\mathsf{R}^{\mathsf{hyp}}$ is range restricted.*

For range restricted clauses, the open branches of a complete derivation tree T constructed by a $\mathsf{R}^{\mathsf{hyp}}$ derivation for N describe Herbrand models in a very simple way. These models are finite for any subclass of the class of range-restricted clause sets which is decidable by $\mathsf{R}^{\mathsf{hyp}}$, for example the class of GF1$^-$ clause sets. Let $[\![B]\!]$ denote the set of positive ground unit clauses in the limit N_∞ of a branch B of a (complete or incomplete) derivation tree T for a clause set N.

Theorem 5 ([11]). *Let N be the clausal form of a GF1$^-$ formula φ, and let N_∞ be the limit of any branch B in an $\mathsf{R}^{\mathsf{hyp}}$ derivation tree with root N. If N_∞ does not contain the empty clause, then $[\![B]\!]$ is a finite (Herbrand) model of N.*

We refer to $[\![B]\!]$ as a (partial Herbrand) model for N represented by the open branch B.

Theorem 6 (Minimal model completeness [6]). *Let N be a satisfiable set of range restricted clauses, and let T be any complete $\mathsf{R}^{\mathsf{hyp}}$ derivation tree constructed from N. For every minimal model H there is an open branch B in T such that $[\![B]\!]$ coincides with the ground atoms in H.*

As GF1$^-$ clause sets are range restricted it follows immediately that for every minimal model H there is an open branch B in a derivation from a GF1$^-$ clause set N such that $[\![B]\!]$ coincides with the ground atoms in H. However, not every model computed with $\mathsf{R}^{\mathsf{hyp}}$ is a minimal model of the input set.

One way to ensure that only minimal models are generated is the use of model constraints. If $[\![B]\!] = \{A_1, \ldots, A_n\}$ is the finite Herbrand model of an open branch B, then let $\overline{[\![B]\!]}$ denote the clause $\neg A_1 \vee \ldots \vee \neg A_n$. The clause $\overline{[\![B]\!]}$ is called a *model constraint*. Suppose B is an open, complete branch in a derivation and we add $\overline{[\![B]\!]}$ to a branch B' distinct from B. If B' would otherwise generate a model $[\![B']\!]$ which is a superset of $[\![B]\!]$, then we will now be able to derive a contradiction using $\overline{[\![B]\!]}$. Because $[\![B]\!]$ may not be a minimal model, it is not enough to add a model constraint to branches which are incomplete with respect to $\mathsf{R}^{\mathsf{hyp}}$ (i.e. branches which can be further expanded by the application of an $\mathsf{R}^{\mathsf{hyp}}$ inference rule). A model constraint also needs to be added to branches complete with respect to $\mathsf{R}^{\mathsf{hyp}}$. In this case, adding the model constraint $\overline{[\![B]\!]}$ to the leaf of a complete branch B' may change the status of B' from complete to incomplete, and it is possible to perform a single additional hyperresolution inference which closes B'. Some additional bookkeeping is necessary to ensure that a model constraint generated by a branch B is propagated only once. We achieve this by introducing the concept of a *finished branch.* A branch is *finished* once its model constraint has been added to all relevant branches of the derivation.

Formally, we extend our calculus by the following *model constraint propagation* rule. We use the notation $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ to refer to the calculus based on factoring, hyperresolution, splitting, and model constraint propagation.

Model constraint propagation: Let B be an open, non-finished branch which is complete with respect to $\mathsf{R}^{\mathsf{hyp}}$. Then add the clause $\overline{[\![B]\!]}$ to all leaves of branches in the derivation tree which are incomplete with respect to $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ or are marked finished. We say the model constraint propagation rule *is applied to B*. Once the rule has been applied, B will be marked *finished.*

The rule ensures that the model constraint is propagated to all relevant branches. Since the model constraint propagation rule can only be applied to branches which are not marked as finished and a branch is marked finished immediately after the model propagation rule has been applied to it, the rule is applied at most once to any branch in the derivation tree.

Theorem 7. *Let φ be a GF1$^-$ formula and let N be the corresponding clause set. Then: (i) Any $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ derivation from N terminates. (ii) φ is unsatisfiable iff all branches in any completed $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ derivation tree with root N are closed.*

Theorem 8. *Let N be the clausal form of a GF1$^-$ formula φ. Then: (i) If $[\![B]\!]$ is the set of positive ground unit clauses in the limit of an open branch in an $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$*

derivation tree from N, $[\![B]\!]$ forms a minimal Herbrand model for N. (ii) $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ generates only minimal Herbrand models for N. (iii) $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ generates all minimal Herbrand models for N, and does so only once.

$\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ is generally sound and complete. For range restricted clauses, the previous theorem is true as well. But $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ is, of course, not a decision procedure for the class of range restricted clauses, for example, it does not terminate on this simple clause set: $\{P(a), \neg P(x) \vee P(f(x))\}$.

The calculus above supports arbitrary branch selection functions and clause selection functions provided that derivations are fair. However, branch selection and clause selection functions have an impact on the performance of an implementation, and developing good strategies is crucial for practical applications.

The minimal model generation procedure of Bry and Yahya [6] can be viewed as a refinement of $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ with an additional rule, the *complement splitting* rule, and with a particular branch selection function which always selects the left-most open branch in a derivation tree.

Complement splitting: $\dfrac{N \cup \{C_1 \vee C_2\}}{N \cup \{C_1, \neg C_2\} \quad | \quad N \cup \{C_2\}}$
where C_2 is a ground clause.

The complement splitting rule can be seen as variant of the folding down rule [17] or as a combination of the cut rule (applied to C_2), clause reduction (replacing $C_1 \vee C_2$ by C_1 in the presence of $\neg C_2$), and subsumption deletion (removing $C_1 \vee C_2$ in the presence of C_2). Complement splitting ensures that the first (and left most) completed open branch B determines a minimal model. The model constraint propagation rule then adds the model constraint $\overline{[\![B]\!]}$ to all leaves of branches to the right of the current branch. Subsequent models generated are always minimal, and only constraints of minimal models are propagated. Hence, there is no need for the marking scheme of $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ which ensures that the model constraint propagation rule is applied only once to a branch.

The Bry-Yahya procedure is sound and complete, and for range restricted clauses the set of generated models is exactly the set of minimal Herbrand models of the input set [6]. Thus clearly, this approach is also applicable to GF1$^-$.

A disadvantage of $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ and the Bry-Yahya procedure are their worst-case space requirement. Let φ be a GF1$^-$ formula of size n and let N be the corresponding set of clauses. Let $\epsilon_2(n)$ denote n^{n^n}. Since we have to maintain a complete representation of the models during the derivation, the space optimisation techniques described in Section 3 cannot be applied here. Ignoring the model constraints, the space required to maintain the essential information on a branch, in other words, the clauses in the leaf of the branch, is bounded by $O(n^2\epsilon_2(n))$ which is also the space required to store a Herbrand model of N, while the number of branches is bounded by $O(2^{\epsilon_2(n)})$. Since a model constraint contains the negation of each atom occurring in a Herbrand model, the space required for a model constraint is again bounded by $O(n^2\epsilon_2(n))$. In the worst case, each branch generates a model constraint which is propagated to all the remaining branches. Thus, in the worst case, a minimal model generation procedure based on $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ stores all the essential information for all the branches in the derivation plus all the model constraints propagated to all branches in space bounded by $O(2^{\epsilon_2(n)}(n^2\epsilon_2(n) + 2^{\epsilon_2(n)}n^2\epsilon_2(n)))$. This gives a triple exponential space bound. Using the Bry-Yahya approach we can discard a branch once it has been closed or has produced a minimal model and its model constraint has been propagated, and only for minimal models do we generate and keep model constraints. This brings the space requirement down to $O(n^2\epsilon_2(n) + \binom{\epsilon_2(n)}{\lfloor\epsilon_2(n)/2\rfloor}n^2\epsilon_2(n))$. This is an improvement over the general $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ procedure, although it is still triple exponential. Likewise, the time complexity of $\mathsf{R}^{\mathsf{hyp}}_{\mathsf{min}}$ derivations is a triple exponential function.

A way of reducing the space requirement of minimal model generation is by adopting the approach of Niemelä [20] which is based on the following observation. Given a (finite) set H of positive ground atoms (or unit clauses) define: $\neg H = \{\neg A \mid A \in H\}$ and $\overline{H} = \bigvee_{A\in H} \neg A$. Let N be a set of clauses and U be the set of all atoms over the Herbrand universe of N. Let H be a finite Herbrand model of N. Then H is a minimal Herbrand model of N iff $MMT(N, H) = N \cup \neg(U - H) \cup \{\overline{H}\}$ is unsatisfiable. This minimality test is called *groundedness test*. Thus, we can use $\mathsf{R}^{\mathsf{hyp}}$ to enumerate all models of a GF1$^-$ clause set N and also use $\mathsf{R}^{\mathsf{hyp}}$ to test each model H for minimality by testing $MMT(N, H)$ for unsatisfiability. This approach has been applied and refined in [2, 5]. A practical problem from our perspective is that these approaches have been described for propositional or ground clause logic only. In this case, the set U, and therefore $U - H$, are always finite. In the case of GF1$^-$, the Herbrand universe of GF1$^-$ clause sets is infinite in general and thus, U and $U - H$ can be infinite sets. However, we observe that in the case of an $\mathsf{R}^{\mathsf{hyp}}$ derivation from $MMT(N, H)$, the clauses in $\neg(U - H)$ have only the effect of deriving a contradiction in any clause set N' derivable from N which contains a positive unit clause not in H. Since H itself is finite, this effect is straightforward to implement.

Procedure 5 defines a minimal model generation procedure MMG using this variant of Niemelä's groundedness test. Like ResolutionProver, MMG operates on the same two sets of clauses $\mathcal{US}$ and $\mathcal{WO}$. Since for the groundedness test we need the initial set of clauses, MMG takes as additional arguments the original sets of usable and worked-off clauses ($\mathcal{IUS}$ and $\mathcal{IWO}$). In its incarnation as groundedness test, procedure MMG requires the Herbrand model H it has to check for minimality as an argument. The last parameter of MMG (i.e. Flag) distinguishes whether MMG operates as the minimal model generator (when the value of Flag is *true*) or as the groundedness test procedure (when the value of Flag is *false*). Applied to a set N of clauses obtained from a GF1$^-$ formula φ, initially $\mathcal{US}$ is the singleton set $\{Q_\varphi(\overline{a})\}$ and $\mathcal{WO}$ is $N - \{Q_\varphi(\overline{a})\}$, and the call MMG($\mathcal{US}$, $\mathcal{WO}$, $\mathcal{US}$, $\mathcal{WO}$, $\emptyset$, true) will print out all minimal Herbrand models of N.

In the procedure MMG, choose' selects an arbitrary clause (according to some heuristic) and returns it together with the set of usable clauses from which the

Procedure 5 Minimal model generation procedure

```
Procedure MMG($\mathcal{US}$, $\mathcal{WO}$, $\mathcal{IUS}$, $\mathcal{IWO}$, $H$, Flag)
local $\mathcal{NEW}$, $C_{mc}$, Given
begin
  repeat
    while ($\mathcal{US} \neq \emptyset$ and ($\perp \notin \mathcal{US}$ or not StackEmpty(Stack))) do
      if ($\perp \in \mathcal{US}$) then
        (Stack, $\mathcal{US}$, $\mathcal{WO}$) := backtrack(Stack, $\mathcal{US}$, $\mathcal{WO}$)
      else
        (Given, $\mathcal{US}$) := choose'($\mathcal{US}$, $\mathcal{WO}$);
        if (Splittable(Given)) then
          $\mathcal{NEW}$ := FirstSplitCase(Given);
          Stack := push(Stack, SecondSplitCase(Given))
        else
          $\mathcal{WO}$ := $\mathcal{WO} \cup$ {Given};
          $\mathcal{NEW}$ := inf(Given, $\mathcal{WO}$);
        if (not Flag and $\exists A : A \in$ PGUC($\mathcal{NEW}$) $\wedge A \notin H$) then
          $\mathcal{US} := \mathcal{US} \cup \{\perp\}$
        else
          ($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$) := ired($\mathcal{NEW}$, $\mathcal{WO}$, $\mathcal{US}$);
          $\mathcal{US} := \mathcal{US} \cup \mathcal{NEW}$
    $C_{mc} := \bigvee_{A \in \mathsf{PGUC}(\mathcal{WO})} \neg A$;
    if (Flag and $\perp \in$ MMG($\mathcal{IUS} \cup \{C_{mc}\}, \mathcal{IWO}, \emptyset, \emptyset$, PGUC($\mathcal{WO}$), false)) then
      print(PGUC($\mathcal{WO}$));
      $\mathcal{US} := \mathcal{US} \cup \{\perp\}$
    else
      return($\mathcal{US}$)
  until (StackEmpty(Stack))
end
```

chosen clause has been removed and PGUC(N) is a function returning the set of all positive ground unit clauses occurring in a clause set N.

We assume that once a minimal model has been generated and printed, we can discard it from the memory. Similar to the Bry-Yahya approach, MMG only needs to store one branch at a time. However, it does not need to store any model constraints. Instead we need some additional space for the minimality test. Again, during the minimality test we only have to store one branch of the derivation and, in addition, the model we test for minimality. For a GF1$^-$ formula of size n, this brings the space requirement down to $O(n^2\epsilon_2(n))$, which is a considerable improvement (by one exponent). An upper bound of the time complexity is $O((2^{2^{\epsilon_2(n)}} + 1)n^2\epsilon_2(n))$, although we believe that improvements of the search strategy during the minimality tests and a closer analysis can improve this bound. So, for GF1$^-$ MMG is a minimal model generator of double exponential space and triple exponential time complexity.

The complexity bounds do not improve under assumptions like the ones we made in Section 3 where we assumed bounds on either the arity of predicate symbols or the number of free variables in subformulae of GF1$^-$ formulae.

References

1. H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philos. Logic*, 27(3):217–274, 1998.
2. C. Aravindan and P. Baumgartner. Theorem proving techniques for view deletion in databases. *Journal of Symbolic Computation*, 29(2):119–147, 2000.
3. L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In *Proc. KGC'93*, vol. 713 of *LNCS*, pp. 83–96. Springer, 1993.
4. P. Baumgartner, P. Fröhlich, U. Furbach, and W. Nejdl. Semantically guided theorem proving for diagnosis applications. In *Proc. IJCAI'97*, pp. 460–465. Morgan Kaufmann, 1997.
5. P. Baumgartner, J. Horton, and B. Spencer. Merge path improvements for minimal model hyper tableaux. In *Proc. TABLEAUX'99*, vol. 1617 of *LNAI*. Springer, 1999.
6. F. Bry and A. Yahya. Positive unit hyperresolution tableaux for minimal model generation. *J. Automated Reasoning*, 25(1):35–82, 2000.
7. H. de Nivelle and M. de Rijke. Deciding the guarded fragments by resolution. To appear in J. Symbolic Computat., 2001.
8. H. de Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic J. IGPL*, 8(3):265–292, 2000.
9. H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. LICS'99*, pp. 295–303. IEEE Computer Society Press, 1999.
10. H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. In *Proc. LICS'99*, pp. 24–34. IEEE Computer Society, 1999.
11. L. Georgieva, U. Hustadt, and R. A. Schmidt. Hyperresolution for guarded formulae. In *Proc. of FTP'2000*, pp. 101–112. Univ. Koblenz-Landau, 2000.
12. E. Grädel. On the restraining power of guards. *J. Symbolic Logic*, 64:1719–1742, 1999.
13. E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. LICS'99*, pp. 45–54. IEEE Computer Society Press, 1999.
14. R. Hasegawa, H. Fujita, and M. Koshimura. Efficient minimal model generation using branching lemmas. In *Proc. CADE-17*, LNAI, pp. 184–199. Springer, 2000.
15. U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. To appear in *J. Automated Reasoning*, 2001.
16. A. Leitsch. Deciding clause classes by semantic clash resolution. *Fundamenta Informatica*, 18:163–182, 1993.
17. R. Letz and G. Stenz. Model elimination and connection tableau procedures. In A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*, pp. 2017–2116. Elsevier, 2001.
18. C. Lutz, U. Sattler, and S. Tobies. A suggestion of an n-ary description logic. In *Proc. DL'99*, pp. 81–85. Linköping University, 1999.
19. J.-J. Ch. Meyer, W. van der Hoek, and B. van Linder. A logical approach to the dynamics of commitments. *Artificial Intelligence*, 113(1–2):1–40, 1999.
20. I. Niemelä. A tableau calculus for minimal model reasoning. In *Proc. TABLEAUX'96*, vol. 1071 of *LNAI*, pp. 278–294. Springer, 1996.
21. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *J. Artificial Intelligence*, 48:1–26, 1991.
22. C. Weidenbach. SPASS: Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*, pp. 1967–2015. Elsevier, 2001.