A Suggestion for an n-ary Description Logic

C. Lutz, U. Sattler, and **S. Tobies**
LuFG Theoretische Informatik, RWTH Aachen, Germany
{clu, sattler, tobies}@cantor.informatik.rwth-aachen.de

1 Motivation

A restriction most Description Logics (DLs) have in common with most Modal Logics is their restriction to unary and binary predicates. To our knowledge, the only DLs that overcome these restrictions and allow for arbitrary n-ary relations are NARYKANDOR [8] and the very expressive DL $\mathcal{DLR}$ [2].

In the field of Modal Logics, there are two generalisations that allow for n-ary predicates: Polyadic Modal Logics [9] and the more expressive Guarded Fragment [1], which was shown to be EXPTIME-complete in [5] and for which a resolution based decision procedure exists [3]. Unfortunately, when extended by operators that are standard in DLs such as number restrictions, features, or transitive roles, this logic becomes undecidable.

In this paper, we present a new DL, $\mathcal{GF}1^-$, that was designed to meet three goals:

- It should allow for n-ary relations,
- "concept" subsumption and satisfiability should be in PSPACE, and
- it should allow the extension with number restrictions and/or transitive roles (without losing decidability).

$\mathcal{GF}1^-$ is a fragment of the *first* Guarded Fragment, which in turn is a fragment of first order logic. Quantified variables in the first Guarded Fragment must be *guarded*, i.e., formulae are restricted to those of the form

$$\exists\overline{x}.(R(\overline{x},\overline{y}) \wedge \phi(\overline{x})) \text{ and } \forall\overline{x}.(R(\overline{x},\overline{y}) \Rightarrow \phi(\overline{x})),$$

where $\overline{x},\overline{y}$ are variable vectors, R is an atom (the so-called guard), and ϕ is a formulae of the first guarded fragment with free variables $\overline{x}$.

$\mathcal{GF}1^-$ is obtained from the first guarded fragment by restricting the way in which variables may occur in guards: The variables of each atom are divided into two parts, and all quantified variables must fill exactly one of these parts. It can easily be seen that $\mathcal{GF}1^-$ extends both $\mathcal{ALCI}$ and polyadic Modal Logics. For example, we can describe unmarried women whose parents are married and who have only married brothers by the following $\mathcal{GF}1^-$-formula.

$$\begin{aligned}&\mathsf{F}(x) \wedge \neg[\exists h.\mathsf{M}(x,h)](\top) \wedge\\ &[\exists m,f.\mathsf{P}(x,m,f)](\mathsf{M}(m,f) \wedge\\ &\qquad[\forall x'.\mathsf{P}(x',m,f)](\mathsf{F}(x') \vee\\ &\qquad\qquad[\exists w.\mathsf{M}(w,x')](\mathsf{F}(w))))\end{aligned}$$

In this preliminary report, we present a PSPACE tableaux algorithm for $\mathcal{GF}1^-$, which we believe can be extended to handle, for example, number restrictions.

2 Preliminaries

In this Section, we introduce $\mathcal{GF}1^-$, explain the syntactic restrictions, define inference problems, and explain why we believe $\mathcal{GF}1^-$ to be a DL.

Definition 1 Let X be a set of variables. The set $\mathsf{free}(\phi) \subseteq X$ denotes the set of variables that occur free in a formula ϕ. For a variable vector $\overline{x} \in X^n$, $\mathsf{var}(\overline{x})$ denotes the set of variables occurring in $\overline{x}$.

Let N_P be a set of *predicate names*. We assume each predicate name to come with an arity $n > 0$ and a "grouping" (i,j) of its parameters with $n = i + j$. A predicate P with grouping (i,j) is written as $P^{(i,j)}$. The set of $\mathcal{GF}1^-$*-formulae* is inductively defined as follows:

1. $P^{(i,j)}(\overline{x})$ is a $\mathcal{GF}1^-$-formula (a so-called *atom*) for a vector $\overline{x} \in X^{i+j}$,
2. $\mathcal{GF}1^-$ is closed under the boolean connectives $\wedge$, $\vee$, and $\neg$,
3. If $P^{(i,j)}$, $Q^{(j,i)}$ are predicates, ϕ is a $\mathcal{GF}1^-$-formula, $\overline{x} \in X^i$, $\overline{y} \in X^j$ are non-empty variable vectors with $\mathsf{var}(\overline{x}) = \mathsf{free}(\phi)$ and $\mathsf{var}(\overline{x}) \cap \mathsf{var}(\overline{y}) = \emptyset$, then

$$\begin{array}{ll}[\exists\overline{x}.P^{(i,j)}(\overline{x},\overline{y})](\phi), & [\exists\overline{x}.Q^{(j,i)}(\overline{y},\overline{x})](\phi),\\ [\forall\overline{x}.P^{(i,j)}(\overline{x},\overline{y})](\phi), & [\forall\overline{x}.Q^{(j,i)}(\overline{y},\overline{x})](\phi)\end{array}$$

are $\mathcal{GF}1^-$-formulae with free variables $\mathsf{var}(\overline{y})$.

The atoms $P^{(i,j)}(\overline{x},\overline{y})$, $Q^{(j,i)}(\overline{y},\overline{x})$ are called *guard-atoms* because they restrict quantification to those $\overline{x}$ satisfying the guard atom.

Let us comment on the syntactic restrictions imposed by this definition:

- For a formula $[\exists\overline{x}.\mathsf{G}(\overline{x},\overline{y})](\phi)$, the free variables of ϕ must be exactly $\overline{x}$. Formulae of the first Guarded Fragment are also subject to this restriction.
- Each (sub-)formula has at least one free variable.
- The variables in a guard atom may only appear in two different patterns. Together with the previous constraint this implies that, for each predicate $P^{(i,j)}$ appearing in a guard, we have $i, j > 0$, and hence no unary predicate may appear as a guard.
- The way in which variables are grouped in a variable vector is not restricted. For example, $[\exists x.P^{(3,2)}(x,x,x,z,y)](\phi(x))$ is a $\mathcal{GF}1^-$-formulae.

The reasons for these restrictions will become apparent in Section 3 and will be further discussed in Section 4. Roughly spoken, without these restrictions, rather complex blocking conditions were necessary to ensure termination of the tableaux algorithm to be devised.

Semantics, interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, and models are defined just as for standard first order logic, where $[\exists\overline{x}.P(\overline{x},\overline{y})](\phi)$ stands for $\exists\overline{x}.P(\overline{x},\overline{y}) \wedge \phi$, and, dually, $[\forall\overline{x}.P(\overline{x},\overline{y})](\phi)$ is to be read as $\forall\overline{x}.P(\overline{x},\overline{y}) \rightarrow \phi$.

A $\mathcal{GF}1^-$-formula ϕ with $\mathsf{free}(\phi) = \{x_1, \dots, x_n\}$ is *satisfiable* iff the (closed) first order formula $\exists x_1, \dots, x_n.\phi$ is satisfiable. A $\mathcal{GF}1^-$-formula ϕ_1 is *subsumed* by a $\mathcal{GF}1^-$-formula ϕ_2 (written $\phi_1 \sqsubseteq \phi_2$) iff $\mathsf{free}(\phi_1) = \mathsf{free}(\phi_2) = \{x_1, \dots, x_n\}$ and $\Phi := [\forall x_1, \dots, x_n.\phi_1](\phi_2)$ is valid. If ϕ_1 is complex, then Φ is clearly not a $\mathcal{GF}1^-$-formula. However, Φ is valid iff

$$\Phi' := [\exists x_1, \dots, x_n.R^{(1,n)}_{\mathsf{new}}(y, x_1, \dots, x_n)](\phi_1 \wedge \neg\phi_2)$$

is not satisfiable, where $R^{(1,n)}_{\mathsf{new}}$ is a predicate that does not occur in ϕ_1 or ϕ_2. Obviously, Φ' is in $\mathcal{GF}1^-$. Hence an algorithm deciding satisfiability of $\mathcal{GF}1^-$-formulae can be used to decide subsumption of $\mathcal{GF}1^-$-formulae.

Is $\mathcal{GF}1^-$ a Description Logic?

In our understanding, the main characteristics of a DL, besides decidability of the relevant inference problems, is *locality*, namely concepts capture properties of an individual that can be determined by looking at the predicates holding for that individual and for those other individuals the first is related to via some predicate. Due to the guarded quantification this is the case for $\mathcal{GF}1^-$.

However, our syntax involves (explicitly quantified) variables—which is not in the spirit of DLs. For example, this makes it easy to distinguish between children of married parents whose mother is a doctor

$$\mathsf{C}(x) \wedge [\exists m, f.\mathsf{P}(x, m, f)](\mathsf{M}(m, f) \wedge \mathsf{D}(m))$$

and those whose father is a doctor

$$\mathsf{C}(x) \wedge [\exists m, f.\mathsf{P}(x, m, f)](\mathsf{M}(m, f) \wedge \mathsf{D}(f)).$$

Nevertheless, one can think of a variable free syntax for $\mathcal{GF}1^-$ that makes these "identification" of variables explicit. The work on a concise variable-free syntax is part of future work.

3 A Tableaux Algorithm for $\mathcal{GF}1^-$

We present a tableaux algorithm which is similar to the standard tableaux algorithm for $\mathcal{ALC}$ [7]. Given a $\mathcal{GF}1^-$-formula ϕ, it tries to construct a model for ϕ. It works on closed formulae—so-called *constraints*—that are obtained from (subformulae of) ϕ by appropriately substituting free variables by constant symbols and applying completion rules.

Definition 2 Let A be a set of constants[1]. If ϕ is a $\mathcal{GF}1^-$-formula with $\mathsf{free}(\phi) = \{x_1, \dots, x_n\}$ and $\{a_1, \dots, a_n\} \subseteq A$, then $\phi[x_1/a_1, \dots, x_n/a_n]$ is obtained by replacing all occurrences of x_i by a_i. For a better readability, we will use $\phi[\overline{x}/\overline{a}]$ as a shorthand for $\phi[x_1/a_1, \dots, x_n/a_n]$.
$\mathcal{GF}1^-$-formulae where all free occurrences of variables are substituted by constants are called *constraints*, and a *constraint system* is a finite set of constraints. Constants $\overline{b}$ are said to be *fresh* for a constraint system S if none of the b_i occurs in S.
All formulae are supposed to be in *negation normal form*, that is, negation is pushed inwards to occur only in front of predicate symbols by making use of de Morgan's laws and the equivalences $\neg[\exists\overline{x}.\phi_1](\phi_2) \Leftrightarrow [\forall\overline{x}.\phi_1](\neg\phi_2)$ and $\neg[\forall\overline{x}.\phi_1](\phi_2) \Leftrightarrow [\exists\overline{x}.\phi_1](\neg\phi_2)$.
A constraint system S is said to contain a *clash* iff $\{P(\overline{a}), \neg P(\overline{a})\} \subseteq S$ for some predicate name P and constants $\overline{a}$, and *clash-free* otherwise. A constraint system S is said to be *complete* iff none of the rules given in Figure 1 can be applied to S.

Given a $\mathcal{GF}1^-$-formula ϕ_0 with $\mathsf{free}(\phi_0) = \{x_1, \dots, x_n\}$ to be tested for satisfiability, the algorithm starts with $S_0 = \{\phi_0[x_1/a_1, \dots, x_n/a_n]\}$ and applies, successively, the completion rules given in Figure 1. It answers "ϕ_0 is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free constraint system, and "ϕ_0 is unsatisfiable" otherwise. The following lemma proves that this tableaux algorithm is sound, complete, and terminating.

[1] which are interpreted as some objects in the interpretation domain—we do not impose the unique name assumption.

1. Conjunction: If $(\phi_1 \wedge \phi_2) \in S$ and $\{\phi_1, \phi_2\} \not\subseteq S$, then
$S \rightarrow \quad S \cup \{\phi_1, \phi_2\}$
2. Disjunction: If $(\phi_1 \vee \phi_2) \in S$ and $\{\phi_1, \phi_2\} \cap S = \emptyset$, then
$S \rightarrow \quad S \cup \{\phi_i\}$ for some $i \in \{1, 2\}$
3. Existential Restriction: If $[\exists x_1, \ldots, x_n.\phi_1](\phi_2) \in S$ and there are no constants $\overline{b}$ such that $\{\phi_1[\overline{x}/\overline{b}], \phi_2[\overline{x}/\overline{b}]\} \subseteq S$, then choose n fresh constants $\overline{b} \in A^n$ and
$S \rightarrow \quad S \cup \{\phi_1[\overline{x}/\overline{b}], \phi_2[\overline{x}/\overline{b}]\}$
4. Universal Restriction: If $[\forall x_1, \ldots, x_n.\phi_1](\phi_2) \in S$ and there are constants $\overline{b}$ such that $\phi_1[\overline{x}/\overline{b}] \in S$ and $\phi_2[\overline{x}/\overline{b}] \notin S$, then
$S \rightarrow \quad S \cup \{\phi_2[\overline{x}/\overline{b}]\}$

Figure 1: The completion rules for $\mathcal{GF}1^-$.

Lemma 3 Let ϕ_0 be a $\mathcal{GF}1^-$-formula with $\mathsf{free}(\phi_0) = \{x_1, \ldots, x_n\}$ and $S_0 = \{\phi_0[x_1/a_1, \ldots, x_n/a_n]\}$. If S is obtained by applying the completion rules in Figure 1 to S_0, then

1. If $\mathcal{I}$ is a model of S and a Rule i can be applied to S, then the Rule i can be applied to S such that it yields S' that is also satisfied by $\mathcal{I}$.
2. If S contains a clash, then S cannot have a model.
3. If S is clash-free and complete, then ϕ_0 has a model.
4. The tableaux algorithm terminates.
5. For a signature N_P of bounded arity and formulae with a bounded number of free variables, the number n_a of constraints containing a constant a is polynomially bounded by $|\phi_0|$.

Proof: 1. This is obvious for Rules 1, 3, and 4. The non-determinism in Rule 2 was the reason for the somewhat complicated formulation in this point, but is obvious as well.

2. Obvious.

3. Given a complete and clash-free constraint system S, a model $\mathcal{I}$ of S can be defined as follows:

$$\Delta^{\mathcal{I}} := \{a \in A \mid a \text{ occurs in } S\}$$
$$\mathcal{I} \models P(\overline{a}) \text{ iff } P(\overline{a}) \in S$$

By induction on the structure of formulae, it can be easily proved that $\mathcal{I}$ satisfies all formulae in S:

- By definition and since S is clash-free, $\mathcal{I}$ satisfies atoms $P^{(n)}(\overline{a}) \in S$ and negated atoms $\neg P^{(n)}(\overline{a}) \in S$.
- Since S is complete, neither Rule 1 nor Rule 2 can be applied. Thus by induction, $\mathcal{I}$ satisfies constraints of the form $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$.
- If $[\exists \overline{x}.\phi_1](\phi_2) \in S$, then $\{\phi_1[\overline{x}/\overline{b}], \phi_2[\overline{x}/\overline{b}]\} \subseteq S$ since S is complete. By induction, $\mathcal{I}$ satisfies $\phi_1[\overline{x}/\overline{b}]$ and $\phi_2[\overline{x}/\overline{b}]$, and hence it satisfies $[\exists \overline{x}.\phi_1](\phi_2)$.
- Suppose $[\forall x_1, \ldots, x_n.\phi_1](\phi_2)$ is in S but not satisfied by $\mathcal{I}$. Hence there is some $\overline{b} \in (\Delta^{\mathcal{I}})^n$ such that $\mathcal{I} \models \phi_1[\overline{x}/\overline{b}]$ and $\mathcal{I} \not\models \phi_2^{\mathcal{I}}[\overline{x}/\overline{b}]$. By the definition of $\mathcal{GF}1^-$-formulae, ϕ_1 is an atom, thus $\mathcal{I} \models \phi_1[\overline{x}/\overline{b}]$ implies $\phi_1[\overline{x}/\overline{b}] \in S$. Since S is complete, Rule 4 cannot be applied to S, hence $\phi_2[\overline{x}/\overline{b}] \in S$. By induction, we have $\mathcal{I} \models \phi_2[\overline{x}/\overline{b}]$, contradicting the assumption.

Since no rule removes constraints, we have $\phi_0 \in S$ and hence $\mathcal{I}$ is a model of ϕ_0.

4. Termination is proved as for standard $\mathcal{ALC}$ tableaux algorithms. First, we need some technical abbreviations: By $\mathsf{depth}(\phi)$, we refer to the maximum quantifier depth in ϕ. We start by showing that the algorithm constructs a tree-like structure: Define, for a constant a, the set $\mathsf{node}(a)$ to be the set of all those constants a' such that a' was generated together with a by Rule 3 or by instantiating the initial formula. $\mathsf{node}(b)$ is said to be a *successor* of $\mathsf{node}(a)$ iff the constants in $\mathsf{node}(b)$ were introduced for some existential constraint containing constants in $\mathsf{node}(a)$. We say that a constraint ϕ is *uni-node* (resp. *bi-node*) iff all constants occurring in ϕ are of the same node (resp. are of two successive nodes).

Claim: Let S be obtained by the application of the completion algorithm to S_0. Then each constraint is either uni-node or occurs as a guard atom and is bi-node.

This can be proved by induction on the number of rule applications. For S_0, this is obvious. The constants in constraints added by the application of Rule 1 or 2 also occur in the constraint which triggered the rule application. Hence the uni-node property is preserved by these rules. Since constraints of the form $[\exists \overline{x}.\phi_1](\phi_2)$ are not guard atoms, Rule 3 is applied to uni-node constraints only. When applied to $[\exists \overline{x}.\phi_1](\phi_2)$, Rule 3 introduces new constants $\overline{b}$ that are, by definition, of the same node, and hence $\phi_2[\overline{x}/\overline{b}]$ is uni-node. Since $[\exists \overline{x}.\phi_1](\phi_2)$ is uni-node and $\overline{b}$ are variables of a single node, the guard atom $\phi_1[\overline{x}/\overline{b}]$ is bi-node. Now let us consider Rule 4 on $\phi = [\forall \overline{x}.\phi_1](\phi_2)$ with $\phi_1[\overline{x}/\overline{b}] \in S$. By induction, the claim holds for $\phi_1[\overline{x}/\overline{b}]$. If $\phi_1[\overline{x}/\overline{b}]$ is uni-node, then clearly $\phi_2[\overline{x}/\overline{b}]$ is uni-node. If $\phi_1[\overline{x}/\overline{b}]$ is a bi-node guard atom $P(\overline{b},\overline{a})$ or $P(\overline{a},\overline{b})$ then, due to the syntax restriction of $\mathcal{GF}1^-$, either $\overline{b}$ was introduced by Rule 3 for an existential constraint containing $\overline{a}$ or vice versa. In both cases, all constants in $\overline{b}$ are of the same node and hence $\phi_2[\overline{x}/\overline{b}]$ is uni-node.

As a consequence of this claim and the fact that constants introduced by Rule 3 are new, the nodes of a constraint system form a tree (where each node is connected via exactly one guard constraint with its predecessor). Termination is then a consequence of the fact that this tree is labelled with a bounded number of constraints (see (a)), built in a monotone way (see (b)), and that it is of bounded depth (see (c)) and width (see (d)):

(a) Uni-node constraints are instantiated sub-formulae of ϕ_0. Since $\mathsf{node}(a)$ never changes after a was generated, there are only finitely many uni-node constraints in which a constant in $\mathsf{node}(a)$ might occur.

(b) No rule removes constraints—each rule adds constraints.

(c) If Rule 3 or 4 adds a constraint $\phi_2[\overline{x}/\overline{b}]$ for some $\phi = [\exists\overline{x}.\phi_1](\phi_2)$ or $\phi = [\forall\overline{x}.\phi_1](\phi_2)$, then $\mathsf{depth}(\phi_2) < \mathsf{depth}(\phi)$. Hence if a node $\mathsf{node}(b)$ is a successor of some $\mathsf{node}(a)$, then the maximum depth of constraints on $\mathsf{node}(b)$ is strictly smaller than the one on $\mathsf{node}(a)$. As a consequence, the depth of the tree (i.e., the maximum number of successive nodes) is bounded by the depth of ϕ_0.

(d) The precondition of Rule 3 ensures that, for each existential constraint on $\mathsf{node}(a)$, at most one successor of $\mathsf{node}(a)$ is generated. Since there are only finitely many constraints on a node, each node has only finitely many successors.

5. (For a signature N_P of bounded arity and formulae with a bounded number of free variables, the number n_a of constraints containing a constant a is polynomially bounded by $|\phi_0|$.) Let N_P be a signature with maximum arity k, let ℓ be an upper bound for the number of free variables, and let K be the maximum of ℓ and k. Then the root node contains at most ℓ constants and each non-root node contains at most $k-1$ constants (since there must be at least one free variable in a guard atom). Taken together, each node contains at most K constants. Hence, for a subformula ϕ of ϕ_0 with $\ell' < k$ free variables and a node $\mathsf{node}(a)$, there are at most $K^{\ell'}$ constraints on $\mathsf{node}(a)$ that are instantiations of ϕ. Since the number of subformulae of ϕ_0 is bounded by the length of ϕ_0, we have that there are at most $K^K \cdot |\phi_0|$ constraints on a node, which is linear in $|\phi_0|$ for a fixed K. ∎

As an immediate consequence of Lemma 3, we have that the algorithm always stops, that a complete and clash-free constraint system defines a model of ϕ_0, and that the rules can be applied in such a way to a satisfiable constraint system that they yield a clash-free and complete constraint system.

Just like for $\mathcal{ALC}$, a naive implementation of this algorithm needs exponential space: It constructs a tree-like structure (of linear depth) that can be exponential in the size of the input. It is easy to see that a trace technique [7] with a reset-restart mechanism similar to the one described in [6] makes it possible to investigate the whole tree while only keeping a single path (of length linear in $|\phi_0|$) in memory at a time. If the number of constraints on each constant is polynomially bounded, this yields a PSPACE algorithm for $\mathcal{GF}1^-$ Hence we have:

Theorem 4 The tableaux algorithm is a decision procedure for the satisfiability of $\mathcal{GF}1^-$-formulae. For a signature N_P of bounded arity and formulae with a bounded number of free variables, it can be implemented to run in polynomial space.

Let us comment on the two bounds we used to obtain a PSPACE result: To have predicates of bounded arity is natural in the field of DL—where the arity is usually restricted to 2. Moreover, this bound is necessary because otherwise exponentially many constraints may be generated for a single node: Consider the following formula Φ_n where G_n is a predicate name of arity $n+1$ and grouping $(1,n)$. The tableaux algorithm—when applied to Φ_n—does not generate a node other than the root node, but generates a constraint for each permutation of the constants $a_1, \ldots, a_n$ (recall that each permutation can be written as a product of transpositions). The number of permutations is exponential in n and hence in the size of Φ_n.

$$
\begin{array}{ll}
\Phi_n := & G_n(x_1, x_1, x_2, \ldots, x_n) \wedge \\
& [\forall\overline{y}.G_n(x_1,\overline{y})](G_n(y_1,y_2,y_1,y_3,y_4,\ldots,y_n)) \wedge \\
& [\forall\overline{y}.G_n(x_1,\overline{y})](G_n(y_1,y_1,y_3,y_2,y_4,\ldots,y_n)) \wedge \\
(1) & \vdots \\
& [\forall\overline{y}.G_n(x_1,\overline{y})](G_n(y_1,y_1,\ldots,y_{i+1},y_i,\ldots,y_n)) \wedge \\
& \vdots \\
& [\forall\overline{y}.G_n(x_1,\overline{y})](G_n(y_1,y_1,y_2,y_3,\ldots,y_n,y_{n-1})) \wedge \\
& \\
(2) & \bigwedge_{1\leq i,j,\leq n} [\forall\overline{y}.G_n(x_i,\overline{y})](G_n(y_j,\overline{y}))
\end{array}
$$

When started for Φ_n, the tableaux algorithm instantiates $x_1, \ldots, x_n$ with fresh constants $a_1, \ldots, a_n$. Rule 1 unfolds the conjunction and makes Rule 4 via constraints (1) applicable to $G(a_1, a_1, \ldots, a_n)$. Successive applications of Rule 4 to this constraint generate $n-1$ constraints $G(a_1, a_2, a_1, a_3, \ldots, a_n)$, $\ldots$, $G(a_1, a_1, a_2, \ldots, a_n, a_{n-1})$—one for each transposition of adjacent constants. To each of these constraints, Rule 4 can be applied via constraints (1), which leads to a set of constraints containing an element for each permutation of $a_1, \ldots, a_n$ that is generated by two successive transpositions. In this second step, the application of Rule 4 to the constraints $G(a_1, a_2, a_1, \ldots)$ creates constraints that do not start with a_1 but with a_2 instead. In order to make these constraints also trigger the application of Rule 4 via constraints (1), the constraint (2) of Φ_n adds constraints with arbitrary first constants. Successive application of Rule 4 will then lead to the generation of constraints $G(a_1, a_{i_1}, \ldots, a_{i_n})$ where $a_{i_1}, \ldots, a_{i_n}$ is a permutation of $a_1, \ldots, a_n$ that is generated by a sequence of transpositions of adjacent elements. Since each permutation can be generated in this manner, the tableaux algorithm consumes exponential space if no bound on the arity of predicates is assumed.

Given a signature of bounded arity, it is natural to also bound the number of free variables: Without bounding this number, it would only be possible to have an unbounded number of free variables on top level (i.e., an unbounded number of constants in the root node), whereas all other nodes would be bounded by the maximum arity of the predicates. Moreover, in most DLs, this number is restricted to 1.

4 Discussion and future work

Motivation for the syntactic restriction of $\mathcal{GF}1^-$: If the grouping of parameters in guard atoms is dropped, it is no longer guaranteed that a simple tableaux algorithm would only generate finite tree structures as described in Lemma 3, and blocking techniques were necessary to prevent the algorithm from generating an infinite branching tree. For example, when applied to the following formula, the tableaux algorithm yields an infinitely branching tree of depth one, whose nodes are connected by an infinite B path. This is due to the use of the parameters in F which does not correspond to the grouping principle. The first parameter in F is duplicated so that it can be (universally) quantified on depth 1 and (existentially) on depth 3. When v is instantiated with a, the tableaux algorithm generates constraints of the form $F(a, a, b_1, b_2), F(a, a, b_2, b_3), F(a, a, b_3, b_4), \ldots$ and $B(b_1, b_2), B(b_2, b_3), B(b_3, b_4)$.

$$
\begin{aligned}
&F(v, v, m, k) \wedge B(v, m) \wedge \\
&[\forall v', m', k'.F(v, v', m', k')](B(m', k') \wedge \\
&\quad [\exists k''.F(v', v', k', k'')](A(k')))
\end{aligned}
$$

Please note that this infinite branching is not caused by using an atom as both guard and non-guard, but solely due to using different patterns of quantified/free variables in the same guard atom F.

Relation with other DLs: The expressive power of $\mathcal{GF}1^-$ is orthogonal to $\mathcal{DLR}$ [2]: On the one hand, $\mathcal{DLR}$ allows for number restrictions and boolean operations on its analogue to guard atoms. On the other hand, $\mathcal{GF}1^-$ allows for formulae with more than one free variable. When compared with NARYKANDOR [8] again $\mathcal{GF}1^-$ falls short in the sense that it does not allow for number restrictions, but NARYKANDOR does not allow for negation and no complete algorithm for subsumption is known.

Extensions: We believe that the tableaux algorithm presented here can be easily extended to number restrictions by adding appropriate extensions of the standard rules for number restrictions and by extending the notion of a clash. Please note that this is not true for the full Guarded Fragment because it becomes undecidable when extended with number restrictions [5]. Another interesting extension is the one with transitive roles (binary relations) and/or axioms. It was shown in [4] that the Guarded Fragment extended with transitive relations, even when restricted to two variables, becomes undecidable. The investigation of these extensions is ongoing work.

References

[1] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *J. of Philosophical Logic*, 27(3):217–274, 1998.

[2] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive query containment in description logics with n-ary relations. In *Proc. of DL'97*, 1997.

[3] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. of LICS-99*, 1999.

[4] H. Ganzinger, Chr. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. In *Proc. 14th IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1999. To appear in LICS'99.

[5] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 1998. to appear.

[6] I. Horrocks, U. Sattler, and S. Tobies. A PSpace-algorithm for deciding $\mathcal{ALCNI}_{R+}$-satisfiability. LTCS-Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998.

[7] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[8] James G. Schmolze. Terminological knowledge representation systems suporting N-ary terms. In *Proc. of KR-89*. Morgan Kaufmann, 1989.

[9] J. van Benthem. Polyadic quantifiers. *Linguistics and Philosophy*, 12(4), 1989.