The Guarded Fragment: Ins and Outs

Carlos Areces, Christof Monz, Hans de Nivelle, and Maarten de Rijke

Abstract

In this short note we discuss several perspectives on the notion of Guarded Fragments (GFs) of first-order logic first introduced by Andréka, van Benthem and Németi. We focus on computational aspects, discussing some applications of GFs together with issues like the design of effective decision methods for specific reasoning tasks and the role of GFs in explaining the general good behavior of modal logics.

Contents

1 Introduction

According to a Spanish saying, "Good things in life come in small packages." First-order logic is beautiful, but — for some purposes — it comes in too big a package. When we are interested in "Logic and Computation," decidability is obviously a first condition to be checked, and first-order logic (FO) is only semi-decidable. It is natural to consider, instead, fragments of first-order logic where decidability does holds.

The pursuit of decidable fragments of FO seems nearly as old as FO itself, and many of the classics in the area are still very much worth reading, including, for instance, Ackermann's [1]. In this note we consider a recent addition to this tradition: the guarded fragment (GF) as introduced by Andréka, van Benthem and Németi [2]. We will argue that GF, and the series of variations to which it has given rise, has some particular additional logical and computational properties which make it stand out among the large number of decidable fragments of FO that have been introduced over the years. In addition to these logical and computational properties, there are also some useability aspects which, in our opinion, give GF a special status.

Put very abstractly, GF forms a basic core upon which new formalisms can be built, addressing specific modeling or computing needs. We believe that this "on-demand" or "engineering" approach to computational logic is a promising one, and, in fact, that it constitutes a natural step which has already been taken in many other fields. In different areas of knowledge representation, for example, much is known about designing specific languages together with ad-hoc decision methods aimed at particular reasoning tasks [9]. The basic picture is the one presented in Figure 1 below.

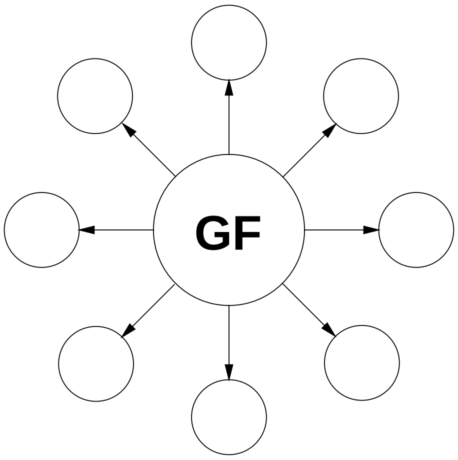

Figure 1: The Guarded Fragment: Ins and Outs

The ideology underlying Figure 1 is best described as follows. We view GF as providing the basic background, while the added "modules" refine the language in response to concrete needs. These "refinements" can take the form of special decision methods for the whole GF, fragments or extensions (see, e.g., [27]), novel logical core results on interpolation, finite model property, expressive power, etc. (see, e.g., [20]), applications "living inside" guarded fragments [3], and many more.

Actually, we would like to push ideology even further: even the central component of the picture displayed in Figure 1 is just a module and can be exchanged by a different fragment, as far as this new fragment enjoys some of the nice features of GF. In general, we advocate a kind of tandem thinking, at multiple levels. The first level is that one should construct the tools for one's specific needs in a general background theory that provides the right perspective. The second level is that the various explorations of syntactically specified (decidable) fragments should go hand in hand with semantic considerations that aim at explaining the good logical and computational behavior that the fragments may display.

The rest of this note is organized as follows. We first provide some background material and basic definitions in Section 2; in Section 3 we discuss some uses of GF; then, Section 4 deals with computational aspects of GF. In Section 5 we discuss the role of GF as a fragment of first-order logic and especially as an explanation of the so-called robust decidability of modal logics. We conclude in Section 6 with some general ideas for future work on GF.

2 Basic Definitions

We start by defining the first-order language we will be dealing with.

A (first-order) relational language is a (first-order) language with equality and without function symbols. We use Rel to denote the countable set of relation symbols, and we assume a fixed arity n for each $R \in \mathsf{Rel}$. For a formula φ, we use $\mathrm{Var}(\varphi)$ and $\mathrm{Free}(\varphi)$ to denote the sets of variables and free variables in φ, respectively. If $\mathcal{M}$ is a first-order model suitable for our relational language, we write $\mathcal{M} \models \varphi(\overline{x})[\overline{a}]$ to denote that the tuple of elements $\overline{a}$ satisfies the formula $\varphi(\overline{x})$ in $\mathcal{M}$.

Let us quickly turn to the guarded fragment now. Intuitively, a relational first-order formula is guarded if all its quantifiers are relativized or "guarded" by atoms in the language. Atoms in guards function as *bridges*, establishing links among the different variables involved in the formula. These bridges keep the structure tight and will be the key tool for establishing results like decidability and the pseudo-tree model property [15].

By specifying the architecture of these 'bridges,' we obtain different kinds of guarded fragments.

Definition 1 (Guarded Formula, [2]) Let $\mathcal{L}$ be a relational language. The *atomic formulas* (or, atoms) of $\mathcal{L}$ are of the usual forms: $v_1 = v_2$, for variables v_1, v_2, and $Rv_1 \dots v_n$, for an n-ary relational symbol $R \in \mathsf{Rel}$, and variables $v_1, \dots, v_n$.

The *guarded formulas* of $\mathcal{L}$ are defined by induction as follows.

1. An atomic formula is a guarded formula.
2. If φ, ψ are guarded formulas, then $\varphi \wedge \psi$ and $\neg\varphi$ are guarded formulas.

3. Let $\overline{v}$ be a finite, non-empty sequence of variables, ψ a guarded formula, and G an atom such that $\mathrm{Free}(\psi) \subseteq \mathrm{Free}(G)$. Then $\exists \overline{v}\,(G \wedge \psi)$ is a guarded formula. G is called the *guard* of the quantifier.

The *guarded fragment* (GF) is the smallest fragment of first-order logic containing all the guarded formulas.

A typical example of a guarded formula is the one expressing symmetry of a relation:

(1) $\forall v_1 v_2\,(Rv_1v_2 \to Rv_2v_1)$.

The formula

(2) $\exists v_2\,(Rv_1v_2 \wedge \psi(v_2) \wedge \forall v_3\,[(Rv_1v_3 \wedge Rv_3v_2) \to \varphi(v_3)])$,

the standard translation of the temporal formula *Until*(φ, ψ), is non-guarded, as the sub-formula $\forall v_3[(Rv_1v_3 \wedge Rv_3v_2) \to \varphi(v_3)]$ is not atomic.

It is not difficult to formalize the connection between modal logics and the guarded fragments. The Standard Translation, ST, maps modal formulas into GF.

Definition 2 (Standard Translation) The translation ST from the modal language into first-order logic over the signature $\langle \{R\} \cup \{P_j \mid p_j \in \mathrm{PROP}\}, \{\}, \{x, y\}\rangle$ is defined by mutual recursion between two functions ST_x and ST_y. Recall that $\varphi[x/y]$ means "replace all free instances of x by y in φ."

$$
\begin{array}{rcl|rcl}
\mathrm{ST}_x(p_j) & = & P_j(x),\, p_j \in \mathrm{PROP} & \mathrm{ST}_y(p_j) & = & P_j(y),\, p_j \in \mathrm{PROP} \\
\mathrm{ST}_x(\neg\varphi) & = & \neg\mathrm{ST}_x(\varphi) & \mathrm{ST}_y(\neg\varphi) & = & \neg\mathrm{ST}_y(\varphi) \\
\mathrm{ST}_x(\varphi \wedge \psi) & = & \mathrm{ST}_x(\varphi) \wedge \mathrm{ST}_x(\psi) & \mathrm{ST}_y(\varphi \wedge \psi) & = & \mathrm{ST}_y(\varphi) \wedge \mathrm{ST}_y(\psi) \\
\mathrm{ST}_x(\Diamond\varphi) & = & \exists y\,(Rxy \wedge \mathrm{ST}_y(\varphi)) & \mathrm{ST}_y(\Diamond\varphi) & = & \exists x\,(Ryx \wedge \mathrm{ST}_x(\varphi))
\end{array}
$$

Observe that the standard translation preserves satisfiability both ways, and that it actually takes modal formulas to guarded formulas. Hence, GF may truly be viewed as a generalization of the modal fragment of FO.

Furthermore, many classical modal systems can be embedded in GF (or a variation) as the conditions characterizing their class of models are often guarded. For example, **KT** (reflexivity) and **KD** (seriality) can be mapped directly into GF. **K4** (transitivity), on the other hand, is not characterized by a guarded formula. And actually, transitivity poses a difficult problem for the guarded approach; see Section 5 for further discussion.

In addition to GF, many further fragments have been introduced. We will now briefly discuss some of them. First of all, in [6] the pairwise (or loosely) guarded formulas were introduced by weakening the conditions on guards.

Definition 3 (Pairwise Guarded Formula) We define pairwise guarded formulas by relaxing clause 3. from the definition of guarded formulas:

3'. If $\overline{v}$ is a finite non-empty sequence of variables, ψ is a pairwise guarded formula, and G is a finite conjunction $\bigwedge_{i=1,\dots,n} G_i$ of atoms such that every variable $v \in \overline{v}$ coexists with every variable in $\mathrm{Free}(\psi) \cup \overline{v}$ in some G_i, then $\exists \overline{v}\,(G \wedge \psi)$ is a pairwise guarded formula.

The *pairwise guarded fragment* (PGF) is the smallest fragment of first-order logic containing all the pairwise guarded formulas.

Notice that the translation of $\mathit{Until}(\varphi, \psi)$ in (2) is indeed pairwise guarded.

Fixed point operators can be added to classical modal logics to obtain dynamic logics or the μ-calculus. Similar extensions can be pursued in the guarded framework [18].

Definition 4 (Guarded and Pairwise Guarded Fixed Point Formulas) The *guarded fixed point logics* μGF and μPGF are obtained by adding to GF and PGF, respectively, the following rule for constructing fixed point formulas.

Let φ be a formula, W a k-ary relation variable that occurs only positively in φ, and let $\overline{x} = x_1, \ldots, x_n$ be a k-tuple of distinct variables. Then we can build the formulas $[\mathrm{LFP}\ W\overline{x}.\varphi](\overline{x})$ and $[\mathrm{GFP}\ W\overline{x}.\varphi](\overline{x})$.

The fixed point semantics for these operators is defined in the usual way.

As an example, the formula

(3) $\forall xy\,(Rxy \to [\mathrm{LFP}\ Wx.(\forall y(Ryx \to Wy))](x))$

is a guarded fixed point formula forcing R to be well founded.

3 Using the Guarded Fragment

In this section we briefly describe what we take to be two of the most promising application areas of GF: natural language technology, and tools for combined logics.

3.1 Natural Language

Several applications in the emerging field of language technology employ theorem proving techniques. Logical approaches to Information Retrieval [29, 30] apply deduction methods to retrieve those documents that suit a user's query from a given collection. Here, the query is translated into a logical formula, and the documents are represented as sets of logical formulas. A document d *suits a query* q if the logical representation Γ of d entails the representation φ of q, that is, if $\Gamma \vdash \varphi$. Admittedly, logical approaches to Information Retrieval are feasible only if the collection of documents is rather small. Appropriate problem domains are, for instance, collections of a few hundred abstracts of papers from scientific journals, or of the technical reports of an institute. In these cases, 'only' a few hundred deductions need to be carried out to retrieve the required information.

Another inference-intensive field within language technology is computational semantics. Here, theorem proving is necessary not only to infer information from the semantic representation of a natural language text, but to construct the semantic representation in the first place, it is necessary to draw hundreds of inferences, as can be seen in the DORIS implementation [7]. The latter is mainly due to the inherent ambiguity of natural language expressions, although it may be possible to disambiguate an expression by excluding some of its readings, because they contradict with contextual information.

It is obvious that the performance (and, hence, the usefulness) of applications in both areas heavily depends on the performance of the tools being used. Although there has been a lot of progress in the development of theorem provers resulting in very fast implementations such as BLIKSEM [26] and SPASS [32], these provers are mainly tuned for mathematical problems. They perform less well if they are applied to problems that require large ontologies, as is typically the case in natural language technology.

Description Logics (DLs) are logics that are built to deal with huge ontologies. Many DLs can be embedded in GF [16, 28]. As a consequence, DLs are less expressive than full first-order logics. McAllester and Givan [22] consider a fragment of Montague Semantics that can be expressed in a DL. Formulas belonging to this fragment have to be quantifier-free, meaning that they do not contain any lambda abstractions. For instance, (4.b), which is the semantic representation of (4.a) belongs to the fragment, but (5.b), representing (5.a), does not.

(4) a. Mary read a book.
    b. $(\ \texttt{Mary}\ \texttt{read}\ (\ \texttt{some}\ \texttt{book}\ ))$

(5) a. Mary read a book that John bought.
    b. $(\ \texttt{Mary}\ \texttt{read}\ (\ \texttt{some}\ (\ \lambda x\ (\ x\ \texttt{book} \wedge (\ \texttt{John}\ (\ \texttt{bought}\ x\ ))))))$

For this quantifier-free fragment, the authors of [22] provide an inference procedure which decides satisfiability of a set of formulas in polynomial time.

Of course, the examples in (4) and (5) also illustrate that, despite their inferential advantages, the expressive power of DLs is much too weak to be used for an exhaustive representation of natural language semantics. But often, we need much less, and more recently, several DLs have been devised to model richer characteristics of natural language, such as ambiguity [21] and plurality [10]. More generally, various applications in language technology simply do not require an in-depth analysis of natural language documents, and DLs seem to allow us to represent the content of a document to a reasonable extent and at the same time they allow us to query a collection of documents efficiently, cf. [23, 24]. Indeed, an important challenge here is to automate the construction of DL-based descriptions that provide a shallow semantic representation of natural language documents.

3.2 Combining Logics

Combining logics for modeling purposes has become a rapidly expanding enterprise that is inspired mainly by concerns about modularity and the wish to join together different kinds of information. As any interesting real world system is a complex composite entity, decomposing its descriptive requirements (for design, verification, or maintenance purposes) into simpler, more restricted reasoning tasks is appealing as it is often the only plausible way forward. It would be an exaggeration to claim that we have a thorough understanding of 'combined methods.' Nevertheless, a core body of notions, questions and results has emerged for an important class of combined logics, and we are beginning to understand how this core theory behaves when we try to apply it outside this particular class.

Does the idea of combining logics actually offer anything new? Some of the possible objections can be justified. Logical combination is a relatively new idea: it has not yet been systematically explored, and there is no established body of results or techniques. Nonetheless, there is a growing body of logic-oriented work in the field, and there are explorations of their uses in AI, computational linguistics, automated deduction, and computer science. An overly critical reaction seems misguided.

Logicians working in the area often focus on so-called transfer issues. Let $\mathbf{L}_1$ and $\mathbf{L}_2$ be two logics — typically, these are special purpose logics with limited expressive power, as it often does not make sense to put together logics with universal expressive power. Let P be a property that the logics may have, say decidability, or axiomatic completeness, and $\oplus$ a specific way of combining $\mathbf{L}_1$ and $\mathbf{L}_2$. The *transfer problem* is this: if $\mathbf{L}_1$ and $\mathbf{L}_2$ enjoy the property P, does their combination $\mathbf{L}_1 \oplus \mathbf{L}_1$ have P as well?

As a rule of thumb, in the absence of interaction between the component logics, we do have transfer; here, absence of interaction means that the component languages do not share any symbols, except maybe the booleans and atomic symbols. Properties that do transfer in this restricted case include the finite model property, decidability, and (under suitable restrictions on the classes of models and the complexity class) of complexity upper bounds [19].

Does combining logics work for actual reasoning systems? That is: can existing tools for the component logics be put together to get tools for combined logics? Obviously, the *re-use* of tools and procedures is one of the key motivations underlying the field. One cannot put together any proof procedures for two logics in a uniform way. First, 'proving' can have different meanings in different logics: (semi-)deciding satisfiability or validity, computing an instantiation, or generating a model. Second, it is not clear where to "plug in" the proof procedure for a logic $\mathbf{L}_1$ into that for a second logic $\mathbf{L}_2$; a proof procedure may have different notions of valuations, or of proof goals.

So what can one do? One way out is to impose special conditions on the calculi that one wants to combine [4]. Alternatively, [14] provide an interesting example by combining efficient propositional decision procedures into a decision procedure for the modal logic $\mathbf{K}$. But another — and far more flexible and powerful — method in the case of modal and modal-like logics, is to use a translation-based approach to theorem proving, by mapping all component logics into a common background logic such as FO. As we have seen in Section 2, many modal logics can be translated, not just into FO, but even into GF. Hence, if the component logics that make up a combined logic can be mapped into GF, and if, moreover, the principles expressing their interaction can be expressed by means of GF formulas, any proof tool for GF is also a proof tool for the combined logic. As a second step, one may, of course, wish to equip such tools with further strategies and refinements to exploit any special features that the combined logic may enjoy — but at least such a move will not have to start from scratch.

4 Computing in the Guarded Fragment

Now that we have had a brief look at some of the areas in which GF is being used, let us turn to concrete, computational issues: what do specialized reasoning tasks in GF cost in principle, and how do we perform them in practice? Below, we review what is know about these matters to date, and we conclude with some open issues.

Decidability of the satisfiability problems for GF and PGF was proved in the original paper on GF [2], using a combination of unraveling and finite quasi-models — which are reminiscent of the filtration methods known from modal and dynamic logics. In [2] it was also shown that GF has the finite model property, i.e., every satisfiable formula in the guarded fragment is satisfiable in a finite model. It is still open whether PGF has the finite model property. But it is known that μGF (and hence μPGF) lack the finite model property: consider the formula in (2) again, and add the following conjuncts:

(6) $\exists xy\, Rxy$ and

(7) $\forall xy\, (Rxy \to \exists x\, Ryx)$

The combined formula forces an infinite R-path which should also be well-founded, and hence acyclic, by (2). This conjunction is only satisfiable in an infinite model.

Complexity results on GF and PGF were provided by Grädel [15]. The satisfiability problems for GF and PGF are complete for deterministic double exponential time. For the sub-fragments that have only a bounded number of variables or only relation symbols of bounded arity, satisfiability is EXPTIME-complete. Furthermore, by a recent result due to Grädel and Walukiewicz, decidability is preserved when moving to the fixed point extensions of GF and PGF; the satisfiability problems for μGF and μPGF are again complete for 2EXPTIME; see [18].

While these are interesting theoretical results, the next important question is: what about practical algorithms for GF? An important line of attack here is provided by studying the so-called *tree model property* which says that if a formula is satisfiable, then it is satisfiable on a tree-like model. Many familiar propositional modal logics enjoy this property, but unlike familiar propositional modal logics, GF and PGF are not restricted to unary and binary predicate symbols, and, hence, one needs to make precise what the appropriate notion of tree-like model is in this setting, but this can be done using notions from graph theory. A PGF formula with m variables is satisfiable only if it has a model of bounded degree such that the Gaifman graph of this model has tree width at most $m+1$ [15].

The tree model property is of crucial importance for the design of practical, tableaux-based or automata-based algorithms. A resolution-based decision procedure for GF without equality is provided by de Nivelle and de Rijke [27]; their method uses ordered resolution, with a non-liftable ordering that is incomplete in general, but complete for GF; to deal with PGF without equality, a non-trivial modification of hyperresolution is needed on top of the ordering refinement. Ganzinger and de Nivelle [13] provide a decision procedure for GF plus equality which is based on ordered paramodulation with selection.

5 The Right Fragment?

The classic reference on decidable fragments of first-order logic is perhaps [1], where the decidability of fragments like the $\forall^*\exists^*$ or $\forall^*\exists\forall^*$ is proved. These fragments are specified in terms of restrictions on quantifier prefixes, variables, or the vocabulary of relation and function symbols. There is a fairly complete understanding of the classical decision problem for such fragments, and for almost all of them, the complexity has also been determined [8].

Modal logics provide an alternative, more "semantically driven" approach to restricted fragments of FO; this is, perhaps, a reflection of the fact that decidability results for modal logics are often proved by model theoretic means. Moreover, modal logics usually determine fragments of first-order logic which possess extremely nice metalogical properties (simple axiomatizations, decidability with low complexity and robust decision methods, interpolation, etc.) which are not standard for "syntactically driven" fragments of FO. So, why are modal fragments so well-behaved? One of the main motivations for introducing GF and PGF in the first place, was, indeed, to account for the good behavior of modal fragments. To appreciate the contribution made by GF and PGF in this respect, it is useful to look back at an earlier answer.

As the standard translation ST (Definition 2) shows, classical modal logic can be mapped into FO^2, the two-variable fragment of first-order logic, a fact first pointed out by Gabbay [11]. For a while it was thought that the essentials of modal fragments were captured by their being parts of finite variable fragments such as FO^2. This is not a satisfactory explanation of the good behavior of modal fragments, however, as finite variable fragments share neither the good logical properties of modal fragments, nor their computational properties. Let us briefly describe what's at stake here, starting with the logical aspects.

It is known that FO^2, and more generally FO^k, have very poor interpolation properties, and hence in this respect, such finite variable fragments have very limited explanatory power. What about GF? The standard tool to investigate the model theory of modal logics is the notion of bisimulation [5]; using this tool one can provide clean characterizations of the expressive power of the logics involved as well as proofs for basic definability, preservation, and, indeed, interpolation results. The appropriate notion of bisimulation for GF and PGF can easily be defined; it closely resembles the notion of potential isomorphisms in first-order logic.

Let Z be a finite subset of a model $\mathcal{M}$; Z is called *live* if it is either a singleton, or there exists a relation R and a tuple X such that $Z \subseteq \{x \mid x = X(i)\}$ and $X \in R^{\mathcal{M}}$. Z is called *packed* if every pair of elements of Z is live.

A *guarded bisimulation* is a non-empty set F of finite partial isomorphisms between two models $\mathcal{M}$ and $\mathcal{N}$ which satisfies the following back-and-forth conditions. Given any $f : X \to Y$ in F

1. for any live $Z \subseteq M$ there is $g \in F$ with domain Z such that g and f agree on the intersection $X \cap Z$;

2. for any live $W \subseteq N$ there is a $g \in F$ with range W such that the inverses g^{-1} and f^{-1} agree on $Y \cap W$.

Pairwise guarded bisimulations are defined in the same way as guarded bisimulations but using packed sets in conditions 1. and 2. Both guarded and pairwise guarded bisimulations can be restricted to sets of a maximum finite cardinality k, thus obtaining respectively k-guarded and k-pairwise guarded bisimulations.

With these tools at hand the result below follows by a standard argument.

Theorem 5 (Characterizations of GF and PGF) *Let φ be any first-order formula. Then φ is invariant for (pairwise) guarded bisimulations iff it is equivalent to a formula in GF (or PGF).*

Returning to the main issue at stake — what about interpolation in GF and PGF? It turns out that matters are far more subtle here than in finite variable fragments. Let us try to give an indication of what's going on.

First, using the notions of bisimulation just introduced, Hoogland and Marx [20] provide a number of counterexamples for Craig interpolation, both for GF and PGF. More precisely, they show the following:

1. There exist sentences $\varphi, \psi \in GF$ using only 3 variables and predicate symbols of arity at most 3 such that $\models \varphi \to \psi$, for which there does not exist an interpolant in GF (in any number of variables).
2. There exist sentences $\varphi, \psi \in PGF$ using only 2 variables and predicate symbols of arity at most 3 such that $\models \varphi \to \psi$, for which there does not exist an interpolant in PGF (in any number of variables).

These certainly seem to be very negative results, and they seem to leave little hope that GF or PGF may be used to explain the good logical behavior of modal fragments. However, a careful analysis of the proofs actually yields a positive result. Basically, the explanation boils down to the fact that, in its full generality, plain Craig interpolation is a property which is not "fair" to ask for in guarded fragments. Guarded fragments share an important characteristic: they restrain the power of classical quantifiers. But it can be argued that guards should indeed *only* be constraints. That is, the language used in guards should be disjoint from the language used elsewhere. This separation of concerns is certainly present in usual modal languages, and if FO fragments are to explain the good logical behavior of modal fragments, it may be appropriate to use similar separations. The following definition implements this idea in a novel definition of the interpolation property.

Definition 6 (Modal Interpolation Property) Let φ be a formula in GF or PGF. Let $\mathsf{Act\text{-}Rel}(\varphi)$ be the set of relations symbols occurring in a guard in φ and $\mathsf{St\text{-}Rel}(\varphi)$ the set of relation symbols occurring in φ but never in a guard position.

We say that a logic $\mathbf{L}$ has the *modal interpolation property* if for all $\mathbf{L}$-formulas φ, ψ such that $\models \varphi \to \psi$, there exists an $\mathbf{L}$-formula I such that

1. $\mathsf{Act\text{-}Rel}(I) \subseteq (\mathsf{Act\text{-}Rel}(\varphi) \cup \mathsf{Act\text{-}Rel}(\psi))$,
2. $\mathsf{St\text{-}Rel}(I) \subseteq (\mathsf{St\text{-}Rel}(\varphi) \cup \mathsf{St\text{-}Rel}(\psi)) \cap \mathsf{Rel}(\varphi) \cap \mathsf{Rel}(\psi)$, and

3. $\models \varphi \to I$ and $\models I \to \psi$.

This version of the interpolation property does hold for guarded fragments, even when restricted to relations of a given fixed arity.

Theorem 7 *Let k be any natural number. The guarded fragment which uses predicate symbols of arity at most k has the modal interpolation property. Hence, GF has the modal interpolation property.*

At present it is still open whether finite variable fragments of GF enjoy the modal interpolation property.

Let us change tack now, and see to which extent GF explains the good computational behavior of modal fragments. First, we need to agree on what we mean by the latter. Vardi [31] has stressed the importance of the *robust* decidability of modal fragments: not only is the modal logic **K** decidable, but it remains decidable when we add temporal constructs, fixed point operators, counting, ... FO^2 is decidable [25], and one might want to consider FO^2 as a candidate for explaining the good computational behavior of modal logic, but many simple extensions of FO^2 are undecidable or even highly undecidable, and, hence, FO^2 certainly can't explain robustness [17]. Moreover, modal logics using operators of arity 2 or higher don't live in FO^2, but in one of its extensions FO^k, where $k \geq 3$—but all of these are undecidable.

It is natural to attempt to explain robustness by means of the guarded fragments which, at least intuitively, encode more of the "modal spirit" in their definition than finite variable fragments. However, GF and PGF aren't quite as robust as basic modal logic: extensions with functionality or transitivity statements immediately yield undecidability, as does the addition of counting quantifiers [15]. In contrast,as we mentioned before, the fixed point extensions μGF and μPGF of GF and PGF, respectively, are decidable [18].

The non-robustness results invite us to reconsider the definition of guarded fragments. As we have seen in our discussion of interpolation, in addition to restraining quantificational power by means of guards, we may also want to make a distinction between action predicates and state predicates. This makes a substantial difference for logical properties, but it remains to be determined what its influence is on computational properties. As a first step in this direction, Ganzinger, Meyer, and Veanes [12] show that GF^2 (GF with only two variables) where, in addition, binary relations may be specified as transitive, is undecidable, but that making the strict separation between action and state predicates and allowing transitivity statements only for action predicates, restores decidability.

6 The Way Forward

In this note we have tried to sketch some of the many different aspects of the guarded fragments. We'd like to stress our main ideology again: to us, guarded fragments form a basic core upon which new formalisms can be built, addressing specific modeling or computing needs. We believe that this "engineering" approach to computational logic is a promising one.

Many questions remain. One of the most urgent ones has to do with the issue of separating action predicates and state predicates raised in Section 5—what are the logical and computational benefits? Will we be able, for instance, to exploit special proof strategies? Experiments seem to suggest that making a rigid distinction along these lines would improve the performance of provers such as Bliksem.

Another line of questions relates to the fact that GF and PGF embed many description logics, as discussed in Section 3, but this embedding is only at the level of concepts—what about some of the familiar reasoning tasks often considered in the setting of description logics? What are their costs?

Finally, and in line with Figure 1, we conclude with a recommendation: one should think of guarded fragments as a central theme from which different soloist can improvise their own melody, or as an object in an object-oriented library from which refined instances can been created. Guarded Fragments constitute the fabric from which hand-tailored languages well suited for particular needs can be cut out.

Acknowledgments. Christof Monz was supported by the Physical Sciences Council with financial aid from the Netherlands Organization for Scientific Research (NWO), project 612-13-001. Hans de Nivelle and Maarten de Rijke were supported by the Spinoza Project 'Logic in Action' at ILLC, the University of Amsterdam.

References

[1] W. Ackermann. *Solvable Cases of the Decision Problem.* North-Holland, Amsterdam, 1954.

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

[3] C. Areces, W. Bouma, and M. de Rijke. Description logics and feature interaction. To appear in *Proc. DL'99.* Linköping, Sweden, 1999.

[4] B. Beckert and D. Gabbay. Fibring semantic tableaux. In *Proc. Tableaux'98*, pages 77–92, 1998.

[5] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, 1983.

[6] J. van Benthem. Dynamic bits and pieces. Technical Report LP-97-01, Institute for Logic, Language and Computation, University of Amsterdam, 1997.

[7] P. Blackburn, J. Bos, M. Kohlhase, and H. de Nivelle. Inference and computational semantics. In H. Bunt and E. Thijsse, editors, *IWCS-3*, pages 5–21. Tilburg, 1999.

[8] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer Verlag, 1997.

[9] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford, 1996.

[10] E. Franconi. A treatment of plurals and plural quantifications based on a theory of collections. *Minds and Machines*, 3(4):453–474, 1993.

[11] D. Gabbay. Expressive functional completeness in tense logic. In U. Mönnich, editor, *Aspects of Philosophical Logic*, pages 91–117. D. Reidel Publishing Company, 1981.

[12] H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. Unpublished.

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

[14] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures. In *Proc. CADE-13*, pages 583–597, 1996.

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

[16] E. Grädel. Description logics and guarded fragments of first order logic. In E. Franconi, G. De Giacomo, R. MacGregor, W. Nutt, and C. Welty, editors, *Proc. DL'98*, 1998.

[17] E. Grädel and M. Otto. On logics with two variables. *Theoretical Computer Science*, to appear.

[18] E. Grädel and I. Walukiewicz. Guarded fixed point logic, 1999. Unpublished.

[19] E. Hemaspaandra. Complexity transfer for modal logic. In *Proc. LICS-94*, pages 164–173, 1994.

[20] E. Hoogland and M. Marx. Interpolation in guarded fragments. Unpublished.

[21] U. Küssner. Description logic unplugged. In E. Franconi, G. De Giacomo, R. MacGregor, W. Nutt, and C.A. Welty, editors, *Proc. DL'98*, 1998.

[22] D. McAllester and R. Givan. Natural language syntax and first-order inference. *Artificial Intelligence*, 56(1):1–20, 1992.

[23] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *Proc. SIGIR-93*, pages 298–307, 1993.

[24] R. Möller, V. Haarslev, and B. Neumann. Semantics-based information retrieval. In *Proc. IT & KNOWS-98*, pages 48–61, 1998.

[25] M. Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 21:135–140, 1975.

[26] H. de Nivelle. *Bliksem User Manual*. Institute for Logic, Language and Computation, University of Amsterdam, 1998.

[27] H. de Nivelle and M. de Rijke. Deciding GF and LGF by resolution. Submitted, 1999.

[28] M. de Rijke. Modal logics and description logics. In E. Franconi, G. De Giacomo, R.M. MacGregor, W. Nutt, and C.A. Welty, editors, *Proc. DL'98*, pages 1–3, 1998.

[29] C. van Rijsbergen. *Information Retrieval*. Buttersworth, London, 2nd edition, 1979.

[30] F. Sebastiani. On the role of logic in information retrieval. *Information Processing and Management*, 34(1):1–18, 1998.

[31] M. Vardi. Why is modal logic so robustly decidable? In *Descriptive complexity and finite models (Princeton, NJ, 1996)*, pages 149–183. Amer. Math. Soc., Providence, RI, 1997.

[32] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER, version 0.42. In *13th International Conference on Automated Deduction, CADE-13*, LNAI. Springer, 1996.