Declarative Programming of Search Problems with Built-in Arithmetic

Eugenia Ternovska and **David G. Mitchell**
School of Computing Science
Simon Fraser University
{ter,mitchell}@cs.sfu.ca

Abstract

We address the problem of providing a logical formalization of arithmetic in declarative modelling languages for NP search problems. The challenge is to simultaneously allow quantification over an infinite domain such as the natural numbers, provide natural modelling facilities, and control expressive power of the language. To address the problem, we introduce an extension of the model expansion (MX) based framework to finite structures embedded in an infinite secondary structure, together with "double-guarded" logics for representing MX specifications for these structures. The logics also contain multi-set functions (aggregate operations). Our main result is that these logics capture the complexity class NP on "small-cost" arithmetical structures.

1 Introduction

Several lines of work in "constraint modelling languages" or "declarative programming for search problems" aim to produce high-level declarative languages for representing combinatorial search problems, together with solvers for applying these languages in practice. Underlying much of this work is a common logical task of *model expansion*. Defined for an arbitrary logic $\mathcal{L}$, the task is:

Model Expansion for logic $\mathcal{L}$ (abbreviated $\mathcal{L}$ MX)
Given: *1.* An $\mathcal{L}$-formula ϕ
2. A structure $\mathcal{A}$ for a part σ of $vocab(\phi)$
Find: an expansion $\mathcal{B}$ of $\mathcal{A}$ that satisfies ϕ.

We call the vocabulary of $\mathcal{A}$ the *instance* vocabulary, and $\varepsilon := vocab(\phi) \setminus \sigma$ the *expansion* vocabulary. In the *combined setting* an instance consists of a structure together with a formula. We focus here on the *parameterized setting*, where a fixed formula for each problem constitutes of a problem specification. An instance is a finite structure, and each expansion of this structure that satisfies the formula is a solution.

Ex. 1 *The following formula ϕ of first order logic constitutes an MX specification for Graph 3-colouring:*

$$\begin{array}{c}\forall x\,[(R(x) \vee B(x) \vee G(x)) \\ \wedge\neg((R(x) \wedge B(x)) \vee (R(x) \wedge G(x)) \vee (B(x) \wedge G(x)))] \\ \wedge\, \forall x \forall y\,[E(x,y) \supset (\neg(R(x) \wedge R(y)) \\ \wedge\neg(B(x) \wedge B(y)) \wedge \neg(G(x) \wedge G(y)))].\end{array}$$

An instance is a structure for vocabulary $\sigma = \{E\}$, i.e., a graph $\mathcal{A} = \mathcal{G} = (V; E)$. The task is to find an interpretation for the symbols of the expansion vocabulary $\varepsilon := \{R, B, G\}$ such that the expansion of $\mathcal{A}$ with these is a model of ϕ:

$$\underbrace{(\overbrace{V; E^{\mathcal{A}}}^{\mathcal{A}},\ R^{\mathcal{B}}, B^{\mathcal{B}}, G^{\mathcal{B}})}_{\mathcal{B}} \models \phi.$$

The interpretations of ε, for structures $\mathcal{B}$ that satisfy ϕ, are exactly the proper 3-colourings of $\mathcal{G}$.

Systems explicitly based on MX for FO (with extensions), are reported in [Mitchell *et al.*, 2006; Wittocx and Marien, 2008]. It is not hard to see that model expansion underlies many other languages for modelling combinatorial problems, including many not explicitly based on logic (see, e.g., [Mitchell and Ternovska, 2008]).

A benefit of formalization in logic is that descriptive complexity theory, the study of the relationship between computational complexity and expressiveness of logics [Immerman, 1999], provides tools for analyzing and controlling the expressive power of modelling languages. For example, Fagin's theorem [Fagin, 1974], states that the classes of finite structures definable in $\exists$SO are exactly those in NP. It is equivalent to say that parameterized FO MX captures NP, so FO MX is a natural formal basis for languages to model NP search problems. Capturing results such as Fagin's theorem provide valuable practical, as well as scientific, information. The fact that FO MX can express every problem in NP assures sufficient expressiveness for a user with an NP search problem. That it can express no problem beyond NP provides assurance of reasonable implementability, for example by polytime grounding (automated reduction to, e.g., SAT).

Remark 1 *Since FO MX can specify exactly the problems that $\exists$SO can, one might ask why we use a new term. Primarily it is because we rarely use pure FO MX other than for expository purposes. We employ a variety of logics related to particular goals or particular practical modelling languages. In this paper we restrict FO formulas to those in a certain guarded form, but also extend these with inductive definitions.*

Formalizing Arithmetic FO MX can express every problem in NP, but when numerical properties are involved numbers must be encoded with collections of abstract domain elements, and arithmetic operations defined by relations over

these encodings. Having "built-in" arithmetic constructs is more natural, and is essential to practical languages.

Arithmetic involves an infinite domain, so limiting the expressive power of a language with arithmetic requires bounding the range of quantified variables and expansion predicates. Imposing such bounds may alter the standard semantics of arithmetic operators, and seems to make expressiveness analysis difficult. Adding built-in numbers can easily produce inadvertent increases in expressive power (see [Mitchell and Ternovska, 2008]). Moreover, when we restrict a language to NP in a direct way, it is not clear if all arithmetic properties in NP can be expressed using the built-in arithmetic. For example, a common method to is to represent numbers with tuples of domain elements. However, this approach provides a much smaller fragment of arithmetic, and a much weaker theorem, than we obtain here.

Almost all declarative languages for modelling combinatorial problems have some form of bounded-domain arithmetic. In many cases, this arithmetic has not been formalized. To our knowledge, capturing of NP with built-in arithmetic has not been shown in any area of declarative programming. Arithmetic in has been formalized in related settings, such as database query languages, but no work we are aware of addresses the concerns of this paper.

This report takes steps toward a logical foundation for constraint modelling languages with built-in arithmetic, and methods for controlling their expressive power. We believe the way we do this formalizes much that is done in practice, but without formalization, in some existing languages. Ultimately, we want to produce practical, fully formalized, languages in which modellers may use built-in arithmetic in the most natural way possible. Further, we want to precisely control the expressive power of these languages, in particular to express exactly the NP search problems, with minimal restrictions on syntax.

Contributions We develop a notion of embedded model expansion with an infinite background structure, in particular for arithmetic structures which include the usual functions on the natural numbers (or integers), as well as aggregates such as those used in SQL and many constraint languages (Sec. 2); We introduce a logic for producing embedded MX specifications, which is a variant of the k-guarded fragment of FO (or FO(ID)), in which upper and lower guards control access to the infinite background domain (Sec. 2); We show that, on classes of "small-cost" structures, where numbers are not allowed to be too large, this logic captures NP (Sec. 3); We generalize this result to an extension of classical logic with inductive definitions to allow poly-size "user-defined" guards (Sec. 4).

2 Embedded MX with Arithmetic

Embedded finite model theory (see [Libkin, 2004]), the study of finite structures whose domain is drawn from some infinite structure, was introduced to study databases that contain numbers and numerical constraints. Rather than think of a database as a finite structure, we take it to be a set of finite relations over an infinite domain.

Def. 1 *A structure $\mathcal{A}$ is* embedded *in an infinite* background *(or* secondary*) structure $\mathcal{M} = (U; \bar{M})$ if it is a structure $\mathcal{A} = (U; \bar{R})$ with a finite set $\bar{R}$ of finite relations and functions, where $\bar{M} \cap \bar{R} = \emptyset$. The set of elements of U that occur in some relation or function of $\mathcal{A}$ is the* active domain of $\mathcal{A}$*, denoted $adom_{\mathcal{A}}$.*

In database research, embedded structures are used with logics for expressing queries. Here, we use them in logics for MX specifications (which are second order queries). The vocabularies for these logics consist of 1) σ, the vocabulary of $\mathcal{A}$, which is our instance structure; 2) ν, the vocabulary of an infinite secondary structure $\mathcal{M} = (U; \bar{M})$, such as the arithmetical structure defined below; and 3) ε, an expansion vocabulary. A formula ϕ over $\sigma \cup \nu \cup \varepsilon$ constitutes an MX specification. The model expansion task remains the same: expand an embedded σ-structure to satisfy ϕ.

To limit the expressive power of logics for MX with infinite secondary structures, we must limit the range of quantified variables and the range of expansion predicates. For this, we use an adaptation of the guarded fragment GF_k of FO [Gottlob *et al.*, 2001]. In formulas of GF_k, a conjunction of up to k atoms acts as a *guard* for each quantified variable.

Def. 2 *The k-*guarded fragment *GF_k of FO is the smallest set of formulas that: 1) contains all atomic formulas; 2) is closed under Boolean operations; 3) contains $\exists \bar{x}(G_1 \wedge \ldots \wedge G_m \wedge \phi)$, provided the G_i are atomic formulas, $m \leq k$, $\phi \in GF_k$, and each free variable of ϕ appears in some G_i.*

Here, $G_1 \wedge \ldots \wedge G_m$ is called the *guard* of $\bar{x}$. Since GF_k is closed under negation, universal quantification can be treated as an abbreviation in the usual way, so universal quantifiers are guarded as in $\forall \bar{x}(G_1 \wedge \ldots \wedge G_m \supset \phi)$.

To limit the range of domain elements that may occur in expansion predicates, we introduce GGF_k, a restriction of GF_k where we require and axiom limiting the range of each expansion predicate.

Def. 3 *The double-guarded fragment $\text{GGF}_k(\varepsilon)$ of FO, for a given vocabulary ε, is the set of formulas of the form $\phi \wedge \psi$, with $\varepsilon \subset vocab(\phi \wedge \psi)$, where ϕ is a formula of GF_k, and ψ is a conjunction of guard axioms, one for each symbol of ε occurring in ψ, of the form*

$$\forall \bar{x}(E(\bar{x}) \supset G_1(\bar{x_1}) \wedge \cdots \wedge G_m(\bar{x_m})),$$

where $m \leq k$, and the union of free variables in the G_i is precisely $\bar{x}$.

For expansion functions, the guard axiom is on the graph of the function, i.e., $\forall \bar{x} \forall y \ (f(\bar{x}) = y \supset \phi(\bar{x}, y))$, where ϕ is a conjunction of atoms.

We call the guards of GF_k, which restrict the range of quantified variables, *lower guards*, and the guard axioms of $\text{GGF}_k(\varepsilon)$ *upper guards*. Initially, we require all atoms providing upper and lower guards to be from the instance vocabulary, so ranges of variables and expansion predicates are explicitly limited to $adom_{\mathcal{A}}$. We later relax this restriction, adding a mechanism for "user-defined" guard relations that may contain elements not in $adom_{\mathcal{A}}$. For convenience only,

we assume that the instance vocabulary always contains the predicate symbol $adom$. Then $adom(x)$ can be used as a guard (upper or lower).

Upper and lower guards provide a logical formalization of the type systems of some existing constraint modelling languages [Mitchell and Ternovska, 2008]. Lower guards correspond to declaring the types of variables, and upper guards to declaring the types of arguments to expansion predicates.

Remark 2 *FO MX corresponds to $\exists$SO, and similarly GGF$_k(\varepsilon)$ MX corresponds to k-guarded $\exists$SO, where all variables, first order and second order, have guards.*

For writing MX specifications for embedded structures, we extend the logic GGF$_k(\varepsilon)$ with vocabulary for a fixed background structure $\mathcal{M}$. We will talk about "GGF$_k(\varepsilon)$ MX specifications with background structure $\mathcal{M}$".

The background structure of interest here is the arithmetical structure which we now describe. (This structure is the same as that used in [Grädel and Gurevich, 1998].) In addition to standard arithmetical operators, it has a collection of *multiset operations*, including max, min, sum and product. For any set R, $fm(R)$ denotes the class of all *finite multisets* over R. Any function $f : U \to U$ defines a multiset $mult(f) = \{\!\{f(a) : a \in U\}\!\}$ over U, the domain of $\mathcal{A}$. A multiset operation (or aggregate) is a function $\Gamma : fm(U) \to U$.

Def. 4 *An* Arithmetical structure *is a structure $\mathcal{N}$ containing at least $(\mathbb{N}; 0, 1, \chi, <, +, \cdot, min, max, \sum, \prod)$, with domain $\mathbb{N}$, the natural numbers, and where min, max, $\sum$, and $\prod$ are multi-set operations and $\chi[\phi](\bar{x})$ is the characteristic function. Other functions, predicates, and multi-set operations may be included, provided every function and relation of $\mathcal{N}$ is polytime computable.*

Our logic for GGF$_k(\varepsilon)$ MX specifications with background structure $\mathcal{N}$ is obtained by extending GGF$_k(\varepsilon)$ with terms constructed from the vocabulary of $\mathcal{N}$, which we now define. As usual, $\phi(\bar{x})$ denotes that $\bar{x}$ contains the free variables of ϕ.

Def. 5 (well-formed terms) *Let τ be the vocabulary $\sigma \cup \nu \cup \varepsilon$ and V a countable set of variables. The set of well-formed terms is the closure of the sets of variables V and constants of τ under the following operations:*

1. *If f is a τ-function of arity n, other than a multiset operation or the characteristic function, and $\bar{t}$ is a tuple of terms of length n then $f(\bar{t})$ is a term.*
2. *If Γ is a multiset operation of ν, $f(\bar{x}, \bar{y})$ a term, and $\phi(\bar{x}, \bar{y})$ a τ-formula in which $\bar{x}$ is guarded, then $\Gamma_{\bar{x}}(f(\bar{x}, \bar{y}) : \phi(\bar{x}, \bar{y}))$, is a term with free variables $\bar{y}$.*
3. *If $\phi(\bar{x})$ is a τ-formula such that $\exists\bar{x}\phi(\bar{x})$ is a k-guarded formula, then $\chi[\phi]$ is a term with free variables $\bar{x}$.*

Multiset operations (case 2) act much like quantifiers, binding the variables $\bar{x}$. Notice that the free variables $\bar{y}$ in ϕ, within a multiset operation term, need not be guarded within ϕ. Their guards are in the formula where the term appears.

Semantics of multiset terms Let $G(\bar{y})$ be the multiset term $\Gamma_{\bar{x}}(f(\bar{x}, \bar{y}) : \phi(\bar{x}, \bar{y}))$. The interpretation of $G(\bar{y})$ on τ-structure $\mathcal{D}$ with valuation $\bar{b}$ for $\bar{y}$ is

$$G^{\mathcal{D}}(\bar{b}) = \Gamma\{\!\{f^{\mathcal{D}}(\bar{a}, \bar{b}) : \mathcal{D} \models \phi[\bar{a}, \bar{b}]\}\!\}. \qquad (1)$$

As usual, $\mathcal{A} \models \phi[\bar{a}]$ means that formula $\phi(\bar{x})$, is true in structure $\mathcal{A}$ when the free variables $\bar{x}$ denote domain elements $\bar{a}$.

The index $\bar{x}$ in the term $\Gamma_{\bar{x}}$ is not needed in the semantic definition (1) – think e.g. of Γ being summation (Σ). For readability, we may omit ϕ when true and write $\Gamma_{\bar{x}}(f(\bar{x}, \bar{y}))$; omit free variables and write $\Gamma_{\bar{x}}(f : \phi)$.

The interpretation of the characteristic function $\chi[\phi](\bar{x})$ on τ-structure $\mathcal{D}$ with valuation $\bar{a}$ for $\bar{x}$ is: $\chi[\phi]^{\mathcal{D}}(\bar{a}) = 1$ if $\mathcal{D} \models \phi[\bar{a}]$ and 0 otherwise. We may write $\Gamma_{\bar{x}}(f \times \chi[\phi])$ instead of $\Gamma_{\bar{x}}(f : \phi)$ when Γ is invariant under multiple occurrences of 0 in the multiset (i.e., $\Gamma(S) = \Gamma(S \cup \{\!\{0, \ldots, 0\}\!\})$), as is the case for Σ and max on $\mathcal{N}$.

Def. 6 *An* embedded GGF$_k(\varepsilon)$ MX specification with secondary structure $\mathcal{N}$ *is a set of GGF$_k(\varepsilon)$ sentences over $\sigma \cup \varepsilon \cup \nu$, with terms as in Definition 5, and the secondary ν-structure is the arithmetical structure of Definition 4.*

In our presentation here, all elements of the active domain are drawn from the arithmetical background structure. All results of the paper generalize to the multi-sorted case, including the case where some domains are not ordered.

Remark 3 *One may imagine it would be easier to use a more restricted arithmetic structure, say without multiset functions. In fact, for our main result (Theorem 1), all the difficulty is present as soon as we have numbers and addition. Moreover, the multiset operations make proving the theorem easier.*

We now give examples of embedded MX specifications with secondary structure $\mathcal{N}$, for search versions of two common optimization problems.

Ex. 2 KNAPSACK: Instance vocabulary $\{O, w, v, b_v, b_w\}$, where O is the set of objects; w is the weight function; v is the value function; b_w is the weight bound; and b_v is the value bound. Expansion vocabulary $\{O'\}$, where O' is the set of selected objects. Upper guard axiom: $\forall x(O'(x) \supset O(x))$. Axioms are $\Sigma_x(w(x) : O(x) \wedge O'(x)) \leq b$ and $k \leq \Sigma_x(v(x) : O(x) \wedge O'(x))$, where $t_1 \leq t_2$ is an abbreviation for $t_1 < t_2 \vee t_1 = t_2$. The lower guard for $O(x) \wedge O'(x)$ is $O(x)$. $\diamond$

Ex. 3 MACHINE SCHEDULING PROBLEM [Hooker, 2000]: We must assign jobs to machines to satisfy constraints on release and due dates and a cost bound. The instance lists jobs, machines, possible start times, the release date and due date for each job, the cost and duration for running each job on each machine, and the cost bound. The instance vocabulary, σ, consists of: $Job(j)$, the set of jobs to be scheduled; $Machine(m)$, the set of machines to perform jobs; $Time(t)$, all possible starting times; $ReleaseDate(j)$, a release date for each job; $DueDate(j)$, a due date for each job; $Cost(j, m)$, cost of doing job j on machine m;

$Duration(j, m)$, the duration of executing j on m; and c, the cost bound. The active domain consists of all time points, costs, due and release dates, durations, jobs and machines. The expansion vocabulary consists of two functions: $Assignment(j)$ maps jobs to machines and $StartTime(j)$ maps jobs to start times.

Upper guard axioms:
$\forall j \forall m\ (Assignment(j) = m \supset Machine(m) \wedge Job(j))$
$\forall j \forall t\ (StartTime(j) = t \supset Time(t) \wedge Job(j))$

Axioms:
$\Sigma_j(Cost(j, Assignment(j))\ :\ Job(j)) \leq c$
$\forall j(Job(j) \supset StartTime(j) \geq ReleaseDate(j))$
$\forall j(Job(j) \supset StartTime(j) + Duration(j) \leq DueDate(j))$
$\forall t\ (Time(t) \supset (\forall m\ (Machine(m) \supset$
$max_j(count_j(\psi(j, m, t))) = 1))$.

In the last axiom, which specifies that at most one job is on a machine at a time, $count_j(\psi(j, m, t))$ is an abbreviation for $\Sigma_j(\chi[\psi(j, m, t)])$, and ψ defines the set of jobs being executed on machine m at time t, that is, $\psi(j, m, t)$ is:

$Job(j) \wedge Assignment(j) = m \wedge\ Time(StartTime(j))$
$\wedge\ StartTime(j) \leq t$
$\wedge\ t < StartTime(j) + Duration(j, Assignment(j))$,

It is easy to see that all axioms are in $\text{GGF}_k(\varepsilon)$. $\diamond$

An optimization version would include the objective function: *minimizing:* $\Sigma_j(Cost(j, Assignment(j)) : Job(j))$.

SQL Examples The following SQL query returns the maximum value in column *k* among the tuples in table *T* that satisfy the Boolean condition *C*: SELECT MAX(*k*) FROM *T* WHERE *C*. It is represented by the multiset operation: $max_{x_k}\{\!\{x_k\ :\ \exists x_1 \ldots \exists x_{k-1} \exists x_{k+1} \ldots \exists x_n T(\bar{x}) \wedge C(\bar{x})\}\!\}$, where $\bar{x} := x_1 \ldots x_n$. The following query returns the number of rows in *T* that satisfy the Boolean condition *C*: SELECT COUNT(*) FROM *T* WHERE *C*. Its representation is: $\Sigma_{\bar{x}}(\chi[T(\bar{x}) \wedge C(\bar{x})])$. We have expressed all other SQL aggregates, but for brevity omit them here.

3 Capturing NP in the Presence of Arithmetic

Here, we present our main result. It can be applied to any language which is a syntactic variant of our language, for example a suitable fragment of ESSENCE. We consider the *decision problem* associated with embedded MX, in the *parameterized setting* (data complexity), where the formula is fixed and instances are finite structures.

The size of an embedded structure $\mathcal{A}$ is the size of its active domain, i.e., $|\mathcal{A}| = |adom_{\mathcal{A}}|$. We define the *cost of* $\mathcal{A}$ to be $cost(\mathcal{A}) = \lceil \log(l) \rceil$, where l is the largest number in $adom_{\mathcal{A}}$. A class $\mathcal{K}$ of embedded arithmetical structures has *small cost* if there is some $k \in \mathbb{N}$ such that $cost(\mathcal{A}) < |\mathcal{A}|^k$, for every $\mathcal{A} \in \mathcal{K}$. This is a generalization of the notion of a metafinite structure with small weights of [Grädel and Gurevich, 1998]. The cost of $\mathcal{A}$ is the size of the binary encoding of its largest number, so small cost structures have encodings that are of size polynomial in their domain size. Small cost structures have no numbers larger than $2^{poly(|\mathcal{A}|)}$.

A class $\mathcal{K}$ of τ-structures is an *embedded spectrum* if there is a first-order sentence ϕ of a vocabulary $\tau' := \tau \cup \varepsilon$ of logic $\text{GGF}_k(\varepsilon)$ such that $\mathcal{D} \in \mathcal{K}$ iff there exists an expansion $\mathcal{D}'$ of $\mathcal{D}$ with $\mathcal{D}' \models \phi$.

Theorem 1 *Let $\mathcal{K}$ be an isomorphism-closed class of small-cost arithmetical embedded structures over vocabulary σ. Then the following are equivalent: (1) $\mathcal{K} \in NP$, (2) $\mathcal{K}$ is an embedded spectrum.*

The proof of Theorem 1 essentially reduces this statement to Fagin's theorem, by considering a class of related structures in which numbers are encoded by relations over (abstract) domain elements. For this, we need enough domain elements to encode the largest number, and this is where the small-cost restriction comes from. We believe it is possible to obtain a similar result under a weaker restriction.

4 User-Defined Guard Relations

So far, the numbers that may occur in a solution for an instance are restricted to those that occur in the instance (that is, $\text{adom}_{\mathcal{A}}$), because expansion predicates have upper guards composed only of instance predicates. There are many problems where this is too restrictive, an obvious example being integer factorization. (We can define many search problems with "new" numbers in solutions, but these numbers must be encoded with elements of the instance structure, rather than appearing directly.) To relax this limitation, we introduce "user-defined guard relations". We now consider specifications consisting of two formulas, D and ϕ. Formula D is over vocabulary $\sigma \cup \delta$, where δ is a set of predicate symbols not in σ, and for each instance structure $\mathcal{A}$, D defines an expansion of $\mathcal{A}'$ that includes the new user-defined guard relations. The active domain of A' will be the union of $adom_{\mathcal{A}}$ and any elements of the defined guard relations. Formula ϕ of $\text{GGF}_k(\varepsilon)$, over vocabulary $\sigma \cup \delta \cup \nu \cup \varepsilon$, defines an embedded model expansion task for each $\mathcal{A}'$. That is, ϕ is such a specification with a larger instance vocabulary $\sigma \cup \delta$.

Informally, D should be a device for defining sets of numbers beyond those in the instance, in the aid of letting ϕ be a more natural axiomatization of the problem than it could be without these extra guard relations. For D to fullfill this role, the relations it defines should be unique (for each instance), and easy to compute. Formally, we require D satisfy the following property.

Property 1 (Good D) *We call D* good *if it defines a total function f_D from embedded σ-structures to embedded $\sigma \cup \delta$-structures and is f_D is polytime computable.*

We may choose to use a syntax for D that is distinct from that of ϕ. Whatever the syntactic form we choose for D, we will require that it is in some logic with the property that, for every allowed D, it is decidable (preferably in polytime) if D is good (satisfies property 1). Thus, we can effectively decide if a specification has appropriate user-defined guard relations.

Capturing NP with User-Defined Guards

Let us denote by $\text{DGGF}_k(\varepsilon, \delta)$ a logic obtained by an extension of $\text{GGF}_k(\varepsilon)$ to allow specification of user-defined guards as just described. A $\text{DGGF}_k(\varepsilon, \delta)$ MX specification $\phi \wedge D$ with background structure $\mathcal{N}$ is defined as in Definition 6, except that it also includes the part D.

Lemma 1 *Let $\phi \wedge D$ be a $\text{DGGF}_k(\varepsilon, \delta)$ MX specification with $\mathcal{N}$, where D is good. Let $\mathcal{K}$ be a small cost class of embedded σ-structures, and $\mathcal{K}'$ be the class of $\sigma \cup \delta$-structures obtained from $\mathcal{K}$ by expanding each structure of $\mathcal{K}$ with the relations defined by D. That is, $\mathcal{K}' = \{f_D(\mathcal{A}) : \mathcal{A} \in \mathcal{K}\}$. Then the following are equivalent: (1) $\mathcal{K}$ is in NP (2) $\mathcal{K}'$ is in NP, (3) $\mathcal{K}$ is an embedded spectrum, (4) $\mathcal{K}'$ is an embedded spectrum,*

A Good D for $\mathcal{N}$. Lemma 1 gives general conditions under which we may capture NP with user defined guard relations. In general, similar conditions will hold for many choices of secondary structure, and for each there will be many choices for the form of D that will satisfy those conditions. We now present one choice for the form of D that satisfies the conditions in the case of arithmetical structures. We wish to choose this form so that easily checkable syntactic conditions are sufficient to ensure a given D is good.

Our guard relations will be defined by induction, using the syntax and semantics of FO(ID), the extension of FO with inductive definitions (see [Denecker and Ternovska, 2008]). Inductive definitions are specified with a rule-based syntax, with arbitrary FO formulas in the bodies, under the 2-valued well-founded semantics. We give an example, and refer to [Denecker and Ternovska, 2008] for details of syntax and semantics. The odd numbers on $\mathcal{N}$: $\{\forall x\ (Odd(x) \leftarrow x = 0 \vee \exists y\ (Odd(y) \wedge x = y + 2))\}$ The defined (intensional) symbols are those in the head, and open (extensional) are the rest. We assume that definitions contain no free variables, and every variable that occurs in the head also appears in the body. Each definition may simultaneously define several relations.

Def. 7 *Say a formula ψ of FO(ID) is in form DEF if:*

1. *ψ is a conjunction of definitions of FO(ID) having a well-founded partial pre-order on definitions such that all open (extensional) symbols are either from σ (the instance vocabulary) or defined by a definition which is strictly smaller in the pre-order. (We may also combine all definitions into one large stratifiable definition.)*
2. *Each definition is either: (a) Of the form $\{\forall\, x\ (G(x) \leftarrow x \leq size(\phi))\}$, where $size(\phi)$ is $\Sigma_x(\chi[adom(x))$, or (b) either positive or stratifiable, and every rule is of the form $\{\ \forall \bar{x}\ \bar{y}\ (G(\bar{x}, \bar{y}) \leftarrow y_1 = t_1(\bar{x}) \wedge \ \ldots\ \wedge\ y_k = t_k(\bar{x}) \wedge \phi(\bar{x})\ \}$, where $k \geq 0$ and $\exists\, \bar{x}\ \phi(\bar{x})$ is guarded, and guards are either from the input structure, or are defined earlier in the pre-order.*

Lemma 2 *It is polytime decidable if a formula is in the form DEF. If D is in form DEF, it is good.*

Theorem 2 *DGGF_k MX with background structures and D in form DEF captures NP on small cost structures.*

Ex. 4 *Suppose we have a search problem on weighted directed graphs, and are interested only in nodes within some particular distance of a given node s. In our MX axiomatization, we may need a guard $P(x)$ which represents this set of nodes. We have a sequence of definitions $(\Delta_1, \Delta_2, \Delta_3)$:*

$$\Delta_1 := \{\forall x\ (within\_bound(x) \leftarrow x \leq size(adom))\}\,,$$

where $adom$ is the active domain of the instance. The guard defined in Δ_1 restricts allowable distances.

$$\Delta_2 := \left\{ \begin{array}{c} \forall x\ (distance(x,0) \leftarrow x = s), \\ \forall x \forall y \forall d \forall d_1 \forall d_2 \\ (distance(y,d) \leftarrow within\_bound(d) \wedge \\ distance(x,d_1) \wedge E(x,y,d_2) \wedge d = d_1 + d_2) \end{array} \right\},$$

which defines the distances to all nodes reachable from s provided they are within the distance bound.

$$\Delta_3 := \{\forall x(P(x) \leftarrow \exists d\ distance(x,d))\}\,,$$

which defines the set of all nodes within the pre-specified distance from the node s.

Adding Inductive Definitions to DGGF_k

In [Mitchell and Ternovska, 2005; Mitchell *et al.*, 2006], we proposed to use FO(ID), rather than just FO, as the basic logic for MX specificying NP search problems. Here, again, we find FO(ID) convenient[1]. A generalization of GF_k to the case with inductive definitions is given in [Patterson *et al.*, 2007]. The logics used in the present paper also may be extended with inductive definitions, and the main results will still hold.

5 Related Work

In database theory, embedded model theory (see [Libkin, 2004]) and metafinite model theory [Grädel and Gurevich, 1998; Grädel, 2007], both extend finite model theory to handle applications with infinite domains. Our development closely follows that of [Grädel and Gurevich, 1998], although we use the embedded setting and the guarded logic defined above.

Embedded Model Theory We have taken the embedded models approach in our work, so some results in the area may be useful, but none so far addresses our immediate needs, such as giving conditions for capturing a complexity class. Much work in the area reduces questions about queries over embedded finite models to questions about normal finite models. Many of these results are restricted to generic queries, but declarative programming axiomatizations are rarely generic. An important result is the natural-domain-active-domain collapse for ∃SO for finite models embedded in an infinite structure $\mathcal{M}$. This holds *if* SO quantification is over subsets of the active domain only, but this is too restrictive for us, because solutions to search problems often involves numbers not contained in the instance.

Metafinite Structures [Grädel and Gurevich, 1998] are sorted structures $\mathcal{D} = (\mathcal{A}, \mathcal{R}, \mathcal{W})$, where $\mathcal{A}$ is a finite primary structure, $\mathcal{R}$ is the secondary structure, and $\mathcal{W}$ is a set

[1] Formally we can do without inductive definitions, since FO MX has the same power as ∃SO. However, expressing transitive closure and many other useful concepts is impractically complex in ∃SO.

of "weight functions" from A^k to R. Typically, $\mathcal{R}$ is a fixed infinite structure, such as the arithmetic structure $\mathbb{N}$ that we also use.

Logics for Metafinite Structures are designed to allow application of methods of finite model theory. These are two-sorted logics, interpreted over combined two-sorted structures. In [Grädel and Gurevich, 1998], the logics contain, in addition to the standard terms, *weight terms* which denote functions from the primary to the secondary part. New weight terms can be built by applying functions of the secondary structure to applications of other weight terms.

To obtain capturing of NP, the authors of [Grädel and Gurevich, 1998] define a notion of metafinite spectrum, a counterpart to the generalized spectra in Fagin's sense, and restrict attention to metafinite spectra of structures with "small weights" i.e., if w is a weight function, and $w(\bar{a}) = s$ then $|s| = poly(|\mathcal{A}|)$. Our "small cost" structures and embedded spectra are generalizations of these concepts.

The work of Grädel and Gurevich to a large degree inspired our work here. However, their requirement of "no quantification over the secondary structure" was too restrictive for our purposes. In [Grädel and Gurevich, 1998], access to the secondary structures is through weight terms only. In natural MX specifications, quantification over elements of background structures is essential. Instead of weight terms of [Grädel and Gurevich, 1998], we allow arbitrary mixed relations, and introduced guarded quantification. Using lower guards was not sufficient – on arithmetical structures, unrestricted metafinite spectra capture the r.e. sets [Grädel and Gurevich, 1998], which would imply the same property for our formalism. Thus, we needed upper guards as well. In our proof of capturing NP, we needed to develop a way to deal with numbers appearing as arguments of expansion predicates, which was not needed in [Grädel and Gurevich, 1998].

Other Related Work The authors of [Cadoli and Mancini, 2006] expressed the view that $\exists$SO is a good mathematical abstraction of practical constraint languages in which to carry out a general study of techniques for reasoning about specifications. They explain that $\exists$SO is expressive enough to *axiomatize* finite arithmetic (e.g. modulo domain size), i.e., *guess* tables for e.g. addition, and reject those not satisfying axioms for addition. Our goal was to formalize *built-in* arithmetic as in most constraint languages. The MX-based IDP system [Wittocx and Marien, 2008] has a language with arithmetic and aggregates, but we are not aware of a formalization of the arithmetic or expressiveness analysis for the language. LPARSE can express NEXP-complete problems. A version of ASP with infinite domains is described in [Heymans *et al.*, 2006]. The authors study satisfiability problem, not model expansion, and decidability and complexity results for that problem are obtained for several variants of ASP (loosely guarded programs and generalized programs similar to Datalog LITE). We do not see a clear correspondence between the guarded logics we use here and the loosely guarded fragment used in [Heymans *et al.*, 2006]. No work we are aware of has presented a framework for search problems with built-in arithmetic where the user is given an assurance of universality for a given complexity class.

Acknowledgments
We are grateful to Brendan Guild, Toni Mancini and anonymous referees for discussions and comments.

References

[Cadoli and Mancini, 2006] M. Cadoli and T. Mancini. Automated reformulation of specifications by safe delay of constraints. *Artificial Int.*, 170(8–9):779–801, 2006.

[Denecker and Ternovska, 2008] M. Denecker and E. Ternovska. A logic of non-monotone inductive definitions. *ACM trans. on computational logic*, 9(2):1–51, 2008.

[Fagin, 1974] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation, SIAM-AMC proceedings*, 7:43–73, 1974.

[Gottlob *et al.*, 2001] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. In *ACM Symp. on Principles of Database Systems (PODS '01)*, 2001.

[Grädel and Gurevich, 1998] E. Grädel and Y. Gurevich. Metafinite model theory. *Information and Computation*, 140(1):26–81, 1998.

[Grädel, 2007] E. Grädel. *Finite Model Theory and Descriptive Complexity*, pages 125–230. Springer, 2007.

[Heymans *et al.*, 2006] Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir. Guarded open answer set programming with generalized literals. In *Proc., FoIKS*, pages 179–200, 2006.

[Hooker, 2000] J. Hooker. *Logic-based methods for optimization: combining optimization and constraint satisfaction*, chapter 19, pages 389–422. Wiley and Sons, 2000.

[Immerman, 1999] N. Immerman. *Descriptive complexity*. Springer, 1999.

[Libkin, 2004] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[Mitchell and Ternovska, 2005] D. Mitchell and E. Ternovska. A framework for representing and solving NP search problems. In *Proc. AAAI'05*, 2005.

[Mitchell and Ternovska, 2008] D. G. Mitchell and E. Ternovska. Expressiveness and abstraction in ESSENCE. *Constraints*, 13(2):343–384, 2008.

[Mitchell *et al.*, 2006] D. Mitchell, E. Ternovska, F. Hach, and R. Mohebali. Model expansion as a framework for modelling and solving search problems. Technical Report TR 2006-24, Simon Fraser University, School of Computing Science, 2006.

[Patterson *et al.*, 2007] M. Patterson, Y. Liu, E. Ternovska, and A. Gupta. Grounding for model expansion in k-guarded formulas with inductive definitions. In *Proc. IJCAI'07)*, 2007.

[Wittocx and Marien, 2008] Johan Wittocx and Maarten Marien. *The IDP System*. KUL, June 2008. www.cs.kuleuven.be/~dtai/krr/software/idpmanual.pdf.