The Graedel-Gurevich Small Cost Condition and Capturing NP in Knowledge Representation Languages

Eugenia Ternovska

School of Computing Science
Simon Fraser University
Canada

June 19, 2017

Based on joint work with David Mitchell, Shahab Tasharrofi

Introduction

Grädel and Gurevich introduced Metafinite Model Theory and considered several applications in Computer Science

We study another application – Constraint solving

The area consists of multiple languages, sometimes not very logic-like (or with nonclassical semantics), with associated solvers

e.g. Constraint languages such as ESSENCE, Zink, Answer Set Programming, NP-Spec, programming in a logic of Inductive Definitions, languages for Planning, Integer Linear Programming, Constraint Satisfaction Problems, various integrated solvers with associated languages etc.

Introduction

Constraint languages have built-in arithmetic and aim at solving computationally hard problems, typically NP-hard, declaratively

Our goal was to understand the principles behind these languages, and behind the use of arithmetic

We claimed that it is important to understand the **expressiveness** of these languages from the descriptive complexity point of view in order to provide computational guarantees to the user:

- the language is complete for a complexity class and
- everything one can specify is solvable within the complexity bounds

The Underlying Task

Since languages vary a lot, it is not easy to study the underlying principles systematically

However there is something fundamental and common in all these formalisms

We identified [2005] Model Expansion (MX) task as one of the main tasks solved in the declarative approaches

Model Expansion: Intuitions

A decision procedure (boolean query) corresponds to a class of structures and can be represented as a formula ϕ in some logic

The Model Checking (MC) task is given τ-structure $\mathfrak{A}$ and formula ϕ in the same vocabulary, check $\mathfrak{A} \models \phi$

In practice, especially in constraint solving, we often *specify input vocabulary* and ask the decision procedure to generate interpretation of the output vocabulary

For example, if ϕ axiomatizes all proper 3-Colourings of all graphs, we may specify that we are going to give graphs on the input and ask the procedure to generate all proper 3-Colourings (or the other way, colours on the input, graphs on the output)

Formally, Model Expansion (MX) task:
Given formula ϕ is some logic $\mathcal{L}$ and structure $\mathfrak{A}$ for a part of *vocab*(ϕ), expand $\mathfrak{A}$ to $\mathfrak{B}$ (*vocab*(ϕ)-structure) so that $\mathfrak{B} \models \phi$

can consider decision (is there an expansion?) as well as search (find an expansion) versions of this task

Complexity-wise, MX is in-between the related tasks of Model Checking and Satisfiability: $\text{MC} \leq \text{MX} < \text{SAT}$

MC: the entire structure is given on the input
MX: a part of the structure is given (at least the domain)
SAT: look for a structure to satisfy ϕ

Relative Complexity of the Tasks (in the Finite)

[Kolokolova, Liu, Mitchell, T: 2010]: comparison of the three tasks

as an example, let's look at FO:

Logic	Model checking		Model expansion		Satisfiability
	Combined	Data	Combined	Data	
FO	PSPACE-c *[Sto74]*	$\equiv_{BIT}$AC^0 *[BIS90]*	NEXP-c *[Var82, MT05]*	$\equiv$NP *[Fag74]*	Und. *[Tra50]*
FO(LFP)	EXP-c *[Var82]*	$\equiv_s$P *[I82,V82,L82]*	NEXP-c	$\equiv$ NP	Und.
FO(ID)	EXP-c*	$\equiv_s$P*	NEXP-c*	$\equiv$ NP *[MT05]*	Und.
FO^k	P-c	AC^0	NP-c *[Var95]*	NP-c,$\not\equiv$ NP	Und$_{(k>2)}$, NEXP-c$_{(k=2)}$, EXP-c$_{(k=1)}$
GF$_k$	P-c *[GO99,GLS01]*	AC^0	NEXP-c*	$\equiv$NP$^*{}_{(k\geq 2)}$, NP-c$_{(k=1)}$	Und.$_{(k\geq 2)}$, 2EXP-c$_{(k=1)}$ *[Gra99]*
RGF$_k$	*n.a.*	*n.a.*	NP-c*	NP-c, $\not\equiv$NP*	*n.a.*
μGF	UP$\cap$ co-UP	P	NEXP-c	NP-c	2EXP-c *[GW99]*
GF$_k$(ID)	EXP	P	NEXP-c	k¿1: $\equiv$NP	k¿1: Und.

In constraint solving, the focus is mostly on Model Expansion

But need arithmetic $\Rightarrow$ Metafinite setting

Meta-Finite Structires [Grädel and Gurevich]

Introduced with the goal to apply the methods of finite model theory in an infinite setting

two-sorted structure $D = (\mathfrak{A}; \mathcal{R}; \mathcal{W})$,

where $\mathfrak{A}$ is a finite primary structure,
$\mathcal{R}$ is the secondary structure;
and $\mathcal{W}$ is a set of weight functions from A^k to R

Typically, $\mathcal{R}$ is fixed & infinite, e.g. $\mathbb{N}$ with arithmetic

Quantification is over the primary domain only

For a capturing NP result: Restrict structures to those with small weights i.e., if $w(a) = s$ then $|s| = poly(|A|)$

Meta-Finite Structires [Grädel and Gurevich]

Arithmetic structures $\mathcal{R}$: contain at least 0, 1, $+$, $\times$, $<$, multiset operations *max*, *min*, Σ (sum), Π (product)

All functions, relations, and multiset operations of $\mathcal{R}$ are such that they are evaluated in polytime

Theorem [Grädel and Gurevich]: Let $\mathcal{K}$ be a class of structures with small weights which is closed under isomorphisms. The following are equivalent:

(i) $\mathcal{K}$ is in NP
(ii) $\mathcal{K}$ is a primary generalized spectrum
(i.e., only the primary part is expanded, not the weight functions)

Arithmetic in Constraint Languages

design a language that allows one to use arithmetic **as if it was unrestricted**, and at the same time does not take us outside of NP

provide a natural way to axiomatize problems: "built-in" operations, not be required to axiomatize them, not modulo n operations, not binary encodings

By looking at practical languages we realized that

- quantification over $\mathcal{R}$ (secondary domain) is essential
- need mixed predicates with arguments ranging over both primary and secondary domains
- want arithmetic functions/relations in the expansion vocabulary – solutions may contain numbers!

Need a capturing NP property

Examples

Integer Factorization

Given two numbers n and c, find a non-trivial positive divisor k of n which is smaller than c. The ideal first-order specification of this problem is as follows:

$$\exists k' \ (k \times k' = n) \wedge k \leq c \wedge k > 1 \wedge k < n.$$

each instance of the problem is described with only a few numbers

quantification over elements of the secondary structure

Examples

An ASP axiomatization of Blocked N-Queens

$$1\{Queen(X, Y) : Row(X)\}1 \leftarrow Column(Y).$$
$$1\{Queen(X, Y) : Column(Y)\}1 \leftarrow Row(X).$$
$$\bot \leftarrow Queen(X, Y), Blocked(X, Y).$$
$$\bot \leftarrow Queen(X_1, Y_1), Queen(X_2, Y_2), |X_1 - X_2| = |Y_1 - Y_2|.$$

built-in arithmetic
numbers in the solution (positions)

Examples

Knapsack

$$\forall x(C(x) \supset \mathit{Item}(x)),$$
$$\Sigma_x(W(x) : C(x)) \leq W_k,$$
$$\Sigma_x(V(x) : C(x)) \geq V_k.$$

uses sum aggregates

polynomially solvable in the value of the input (not the size)

Examples

Quadratic Residues

Given three numbers n, a and c, find the modulo-n square root of a which is smaller than c, i.e., a number s such that $s^2 \equiv a \ (\textbf{mod}\ n)$ and $s < c$

$$0 \leq s \wedge s \leq c \wedge s < n \wedge \exists k \ (0 \leq k \wedge k < n \wedge s \times s = k \times n + a).$$

uses only a few numbers, is NP-complete

quantification over elements of the secondary structure

Formalizing Arithmetic: Embedded MX

A Database concept: structure $\mathfrak{A} := (\mathbb{N}; R_1^{\mathfrak{A}}, \dots R_n^{\mathfrak{A}})$
embedded in an infinite background structure
(e.g. $N := (\mathbb{N}; 0^N, succ^N, \leq, min, max, \dots)$

Relations $R_1^{\mathfrak{A}}, \dots R_n^{\mathfrak{A}}$ must be finite

Example: Company database, employees, salaries.
Queries use $+, -, \Sigma, \leq$ etc.

Active domain *adom*:
union of elements appearing in all relations

Embedded MX: the same as before

$$\overbrace{\underbrace{(\mathbf{U}; R_1^{\mathfrak{A}}, \dots R_n^{\mathfrak{A}}}, E_1^{\mathfrak{B}}, \dots, E_m^{\mathfrak{B}})}^{\mathfrak{B}} \models \phi$$

but $\mathbf{U}$ is infinite

Restricting Expansions: Upper Guards

Need weight functions and mixed predicates in the output (expansion) vocabulary (just primary spectra is too restrictive)

Problem: On arithmetic structures, unrestricted metafinite spectra capture the r.e. sets [Grädel and Gurevich]

$\Rightarrow$ limit range of expansion (i.e., solution) predicates

1) Add upper guard axioms

$$\forall x_1, \ldots \ \forall x_7 \ (E(x_1, \ldots, x_7) \ \rightarrow \ G_1(x_1, x_3) \wedge \ldots \wedge G_4(x_3, x_5, x_7)).$$

Each upper guard axiom involves at most k guards

Quantification over Secondary Domain

"No quantification over the secondary domain" is too restrictive

2) Require all quantification over the secondary domain to be guarded in the sense of k-guarded fragment GF_k [Gottlob et al]
Require all quantification to be guarded

- $\forall \bar{x}\ (G(\bar{x}) \Rightarrow \phi(\bar{x}))$ instead of $\forall \bar{x}\ \phi(\bar{x})$,
- $\exists \bar{x}\ (G(\bar{x}) \wedge \phi(\bar{x}))$ instead of $\exists \bar{x}\ \phi(\bar{x})$

where $G(\bar{x})$ is a conjunction of up to k atomic formulas of instance vocabulary e.g. $G_1(x_1, x_3) \wedge G_2(x_3) \wedge G_3(x_2, x_3)$ jointly guarding $\bar{x}$

We call G_1, G_2, G_3, guards for the corresponding quantifier

A limited fragment (unary guards) corresponds to types

Double Guarded Fragment GGF_k

In [T,Mitchell:09] we introduced

a double-guarded fragment $GGF_k(\varepsilon)$:

Require both lower and upper guards for the output vocabulary ε

All lower & upper guards must be either

- given by the instance structure or
- poly-time constructable (we use stratifiable inductive definitions with some restrictions)

Capturing NP with Arithmetic

Recall: arithmetic structure $\mathcal{N}$ [Grädel, Gurevich] contains at least

$$(\mathbb{N};\quad 0, 1, \chi, <, +, \times, \mathit{min}, \mathit{max}, \sum, \prod, \dots)$$

may include other functions, predicates, and multi-set operations, provided polytime computable

the exact choice of operations is very important

Small Cost structures: no input number is too large:

$\mathit{value}(M) \leq 2^{\mathit{poly}(|\mathit{adom}|)}$, $\quad M$ is the largest number in adom

Theorem [T,Mitchell'09] Let $\mathcal{K}$ be an isomorphism-closed class of small-cost arithmetical embedded structures. Then

$\mathcal{K} \in$ NP iff
all structures in $\mathcal{K}$ are expandable to satisfy some ϕ of $\mathrm{GGF}_k(\varepsilon)$

Limitations

1. Cannot capture all of NP (small cost structures only)

Given: a, where $a \in \mathbb{N}$ Find: b_1, b_2
Such that: $b_1 \times b_2 = a$ and b_1, b_2 are prime numbers, $\neq 1$

domain size is 1, the value of input number a is unlimited in terms of domain size

2. Polysize guards are too limiting: cannot represent some problems naturally

$$\exists x \ (\underline{G}(x) \wedge \mathit{Output}(x) \wedge x = \Sigma_y(\ y \ : \ \mathit{Knapsack}(y))$$

cannot put a polysize guard there: 2^n distinct sums

Practical Languages

Our formalization in GGF_k reflects much that is done in practice (but without formalization) in some existing languages (e.g. ASP).

Note: no input value exceeds $2^{poly(|adom|)}$

this small cost condition is present in actual systems!
(the analysis was quite involved)

We show how to eliminate it and capture NP unconditionally

New Logic PBINT

Overcomes the small cost limitation

the exact choice of guards and arithmetic operations is important

before: $(\mathbb{N};\ \ 0, 1, \chi, <, +, \times, \mathit{min}, \mathit{max}, \sum, \prod, \dots)$ plus other polytime operations
now: $(\mathbb{N};\ \ 0, 1, <, +, \times, ||\ ||, \dots)$ plus other polytime operations

where $||x||$ returns the size of binary encoding of x,
i.e., $||x|| = 1 + \lfloor \mathit{log}_2(x+1) \rfloor$

the language has expansion functions

(our results also hold for integers $\mathbb{Z}$)

PBINT Guards

1. **Instance Guards** are instance predicates (including $adom_{\mathfrak{A}}$) interpreted by the instance (input) structure $\mathfrak{A}$

2. **Polynomial Range Guards** are relations of the form $poly_1(SIZE) \leq x \leq poly_2(SIZE)$ where
   $SIZE$ is the number of bits to encode the input structure $\mathfrak{A}$

3. **PBINT Guards** are relations of form $||x|| \leq poly(SIZE)$ where $poly(SIZE)$ is a polynomial

   exponential size: $||x|| \leq SIZE$ is equivalent to $x \leq 2^{SIZE-1} - 1$, exponential in the value of $SIZE$

inductively definable guards can be added

Logic PBINT

Upper Guards

- regular guards (1,2) allowed everywhere
- PBINT guards (exp. size) allowed for the outputs of functions

$$\forall \bar{x} \forall y \ (f(\bar{x}) = y \Rightarrow (G_I(\bar{x}) \wedge G_O(y) \vee y = \mathit{default}))$$

Lower Guards

- regular guards (1,2) allowed everywhere
- PBINT guards (exp. size) allowed for $\exists$

Example: Factorization

Given a number n, find some nontrivial factor p of n.

SIZE is the number of bits to encode the input structure $\mathfrak{A}$

Now, the axiomatization is:
$p > 1 \wedge p < n \wedge \exists q \; (||q|| \leq \mathit{SIZE} \wedge p \times q = n)$.

Main Result

Theorem [Tasharrofi,T'10] Let $\mathcal{K}$ be an isomorphism-closed class of compact arithmetical embedded structures. Then

$\mathcal{K} \in$ NP iff
all structures in $\mathcal{K}$ are expandable to satisfy some embedded PBINT specification ϕ

Proof: use Cook-Bellantoni theorem about capturing PTIME computable functions

and Fagin's characterization of NP by $\exists$SO

Conclusions

We extended the Metafinite setting to Constraint languages

- Our first formalism: GGF_k [T,Mitchell'09]
  - captures NP under small cost condition
  - corresponds to how arithmetic is handled in practical languages (e.g. Answer Set Programming (ASP), the IDP system, ...)
  - some common problems with numbers are not expressible naturally (integer factorization) – binary encodings might be needed
- New logic PBINT [Tasharrofi,T], an idealized declarative language with arithmetic that overcomes those limitations