Guarded Constraint Models Define Treewidth Preserving Reductions

David Mitchell(✉)

Simon Fraser University, Vancouver, Canada
mitchell@cs.sfu.ca
https://www.cs.sfu.ca/~mitchell

Abstract. Combinatorial problem solving is often carried out by reducing problems to SAT or some other finite domain constraint language. Explicitly defining reductions can be avoided by using so-called "model and solve" systems. In this case the user writes a declarative problem specification in a constraint modelling language, such as MiniZinc. The specification implicitly defines a reduction, which is implemented by the constraint solving system. Unfortunately, reductions can destroy useful instance structure, such has having small treewidth. We show that reductions defined by certain guarded first order formulas preserve bounded treewidth. We also show such reductions can be executed automatically from problem specifications written in a guarded existential second order logic (∃SO) by simple grounding or "flattening" algorithms. Many constraint modelling languages are essentially extensions of ∃SO, and this result applies to natural, useful, fragments of these languages.

Keywords: Constraint modelling language · Reduction · Treewidth

1 Introduction

Application of solvers for finite-domain constraint languages, such as FlatZinc and propositional CNF formulas, requires defining an "encoding", which formally is a reduction from the problem of interest to the target solver language. The exact choice of reduction is important to performance in practice, and considerable time is sometimes spent to find a "good" one. It is often observed that some reductions destroy potentially useful instance structure. The formal study of instance structure in constraint solving goes back at least to Freuder's paper [9] which showed that instances of constraint satisfaction problems (CSPs) having bounded treewidth can be solved in polynomial time. More recently, Samer and Szeider gave a detailed study [19], of conditions under which fixed parameter tractability of CSPs follows from bounded treewidth.

Constraint modelling languages and the solving systems that support them eliminate the need to define a reduction explicitly. Users of these "model and solve" systems write a high level declarative description of their problem, and send that together with a problem instance to the system. Almost all existing

© Springer Nature Switzerland AG 2019
T. Schiex and S. de Givry (Eds.): CP 2019, LNCS 11802, pp. 350–365, 2019.
https://doi.org/10.1007/978-3-030-30048-7_21

systems mapping this pair to a single expression in a "flat" language, which has no quantifiers and limited nesting of operators. For example, the MiniZinc system [16] has several options for this flat language including FlatZinc and propositional CNF formulas. For any problem specification S for a problem P, this map is a reduction (often but not always polynomial time) from problem P to the flat language. This reduction is defined by a combination of the specification and the "flattening" or "grounding" algorithm of the system. The user has some control over this reduction in that they can choose among many possible ways to write S, an activity sometimes called "modelling".

This leads to asking under which conditions the reductions implemented by model-and-solve systems could preserve desirable instance structure. Here we consider the case of treewidth, a widely studied structural measure of "tree-likeness" which has produced many tractability results. In particular, we establish sufficient conditions on S under which an instance I of our problem P is mapped to a CNF formula Γ, such that the treewidth of Γ is bounded by the treewidth of P.

We denote the treewidth of instance I by $\mathrm{TW}(I)$. We say a reduction f between problems is *bounded treewidth preserving* (or just treewidth preserving) if there is a function g, depending only on $\mathrm{TW}(I)$, such that, for every problem instance I, $\mathrm{TW}(f(I)) \leq g(\mathrm{TW}(I))$. We allow the treewidth of the image of I to be larger than that of I, but it must not depend on the size of I. We are interested in when the reduction implemented by a model-and-solve system is treewidth preserving.

To study this question formally, we require a formally defined and sufficiently simple specification. We adopt ∃SO, the existential fragment of classical second order logic, as an abstract constraint modelling language. Many actual constraint modelling languages are essentially extensions of ∃SO with arithmetic and other features which are convenient for modelling or specifying problems in practice. By Fagin's Theorem [7] ∃SO can define exactly the problems in the complexity class NP, which seems like a reasonable basis for an initial formal study.

1.1 Contributions

1. We define a family of *guarded reductions*, reductions defined by formulas of first order logic (FO) related to the Packed Fragment of FO, and show that guarded reductions preserve bounded treewidth.
2. We show that, from a "specification" formula Ψ in ∃SO, we can obtain a FO reduction from the NP problem defined by Ψ to SAT.
3. We define a family of guarded ∃SO formulas, also based on the Packed Fragment of FO. We show that basic grounding or flattening algorithms implement a reduction to SAT that is treewidth preserving when the specification is guarded. More precisely, the indicence treewidth of the CNF formula produced is bounded by a polynomial of the treewidth of the problem instance.
4. We show that from a guarded ∃SO specification formula Ψ, we can algorithmically obtain an explicit guarded FO reduction from Mod Ψ, the class of models of Ψ, to SAT. Proofs are sketched due to space limitations.

Guarded specifications are very natural, and occur frequently in practice. The essential idea behind the guardedness property is that quantification should be relativized to an input relation. For example, for a problem in which the input is a graph G, a constraint of the form "for every edge e in G ....." is guarded.

The point here not to solve instances of bounded treewidth, but to obtain reductions which apply to all instances and behave well on those with small treewidth. This behaviour is also relevant to instances which are "almost" of small treewidth. A treewidth preserving reduction to SAT does give (in theory) an efficient algorithm for instances of small treewidth. An efficient algorithm for small treewidth instances, in contrast, does not automatically give us a treewidth preserving reduction.

1.2 Organization

Section 2 defines our basic notation regarding structures, associated graphs, and formulas. Sections 3 and 4 are largely expository, giving required background in FO transductions and reductions. Section 5 defines our guarded reductions and gives the proof that guarded reductions preserve bounded treewidth. Section 6 shows how to obtain FO reductions from ∃SO specifications. Section 7 defines guarded specifications and shows that they induce treewidth preserving reductions, and in particular guarded FO reductions. Section 8 concludes with a summary, discussion of related work, etc.

2 Formal Preliminaries

Problems Are Classes of Structures. A decision problem is taken as an isomorphism-closed class of finite relational structures. This view is standard in descriptive complexity theory, and arguably should be used more generally: it is usually more natural to view a graph property as a set of graphs than as a set of strings encoding graphs.

Logic and Notation. We assume the reader is familiar with the syntax and standard model-theoretic semantics of classical logic. In this section we set out our notation and terminology and also give some examples that will aid our exposition later.

A relational vocabulary is a tuple of one or more relation symbols $\bar{R}$. Each symbol R has an arity $\mathrm{ar}(R)$. A structure $\mathcal{A}$ for vocabulary τ (or τ-structure), is a tuple $(A, \bar{R}^{\mathcal{A}})$ consisting of a nonempty universe or domain A and a relation $R^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R)}$ for each relation symbol $R \in \bar{R}$. The relation $R^{\mathcal{A}}$ is called the interpretation or denotation of R in $\mathcal{A}$. Many authors allow constant symbols in relational vocabularies. Our results would apply also in this case, but we do not include them for simplicity. The size of a structure is the cardinality of its universe. Our structures are all finite, and by default the domain of a size-n structure $\mathcal{A}$ is $A = [n] = \{1, \ldots, n\}$.

Example 1. Let τ_{PL} be the vocabulary $\tau_{\mathrm{PL}} = (\mathrm{Form}, \mathrm{SubF}, \mathrm{Atom}, \mathrm{And}, \mathrm{Or}, \mathrm{Not})$ where $\mathrm{Form}, \mathrm{SubF}, \mathrm{Atom}$ are monadic (unary) and $\mathrm{And}, \mathrm{Or}, \mathrm{Not}$ are binary. A τ_{PL}-structure represents a set of formulas of propositional logic: $\mathrm{SubF}(x)$ means x is a subformula; $\mathrm{Form}(x)$ means x is a formula but is not a proper subformula of another (so $\mathrm{Form}^{\mathcal{A}}$ is the set of formulas in $\mathcal{A}$); $\mathrm{Atom}(x)$ means x is an atomic formula; $\mathrm{And}(x, y)$ means x is a conjunction, and y is one of its conjuncts; $\mathrm{Or}(x, y)$ is dual to And; $\mathrm{Not}(x, y)$ means x is the negation of y.

Example 2. The set $\{(q \wedge \neg r), (p \vee \neg t)\}$ of propositional formulas may be represented by τ_{PL}-structure $\mathcal{A}$ where we number subformulas (e.g., by a pre-order traversal of the formula parse trees). So $A = [8]$ and $\mathrm{Form}^{\mathcal{A}} = \{1, 5\}$, $\mathrm{SubF}^{\mathcal{A}} = [8]$; $\mathrm{Atom}^{\mathcal{A}} = \{2, 4, 6, 8\}$; $\mathrm{And}^{\mathcal{A}} = \{(1, 2), (1, 3)\}$; $\mathrm{Or}^{\mathcal{A}} = \{(5, 6), (5, 7)\}$; $\mathrm{Not}^{\mathcal{A}} = \{(3, 4), (7, 8)\}$;

For first order (FO) formula ϕ, we denote by $\mathrm{free}(\phi)$ the set of free FO variables in ϕ, and write $\phi(\bar{x})$ to indicate that the free variables of ϕ are among those in tuple $\bar{x}$. For simplicity we assume all bound variables are distinct. If $\mathcal{A}$ is a τ-structure, $\bar{a} \in A^k$ and $\phi(\bar{x})$ a τ-formula with k free variables $\bar{x}$, we write $\mathcal{A}, \bar{a} \models \phi(\bar{x})$ to say that if the variables $\bar{x}$ in ϕ denote the elements of $\bar{a} \in A^k$ then ϕ is true in $\mathcal{A}$. We write $\phi(\bar{x})^{\mathcal{A}}$, or just $\phi^{\mathcal{A}}$, for the relation defined by ϕ in $\mathcal{A}$. That is, if ϕ has k free variables, $\phi^{\mathcal{A}} = \phi(\bar{x})^{\mathcal{A}} = \{\bar{a} \in A^k \mid \mathcal{A}, \bar{a} \models \phi(\bar{x})\}$. We write $\mathrm{Mod}\ \phi$ for the class of all finite models of a formula ϕ.

Let $\tau = (R_1, \ldots R_m)$ be a vocabulary, $\mathcal{A} = (A, R_1^{\mathcal{A}}, \ldots R_m^{\mathcal{A}})$ a τ-structure, and $(S_1, \ldots S_n)$ a tuple of relation symbols not in τ. If $\mathcal{B} = (A, R_1^{\mathcal{A}}, \ldots R_m^{\mathcal{A}}, S_1^{\mathcal{B}}, \ldots S_n^{\mathcal{B}})$ is a structure for $\tau' = (R_1, \ldots R_m, S_1, \ldots S_n)$, then we call $\mathcal{A}$ the τ-reduct of $\mathcal{B}$ and $\mathcal{B}$ an expansion of $\mathcal{A}$ to τ'.

Graphs and Treewidth of Structures.

Definition 1 (Gaifman graph). *The Gaifman graph of a relational structure $\mathcal{A}$ is the graph $G(\mathcal{A}) = (A, E)$ with vertex set A and $(a, b) \in E$ if and only if there is a tuple in some relation of $\mathcal{A}$ containing both a and b.*

Example 3. The Gaifman graph of $\mathcal{A}$ from Example 2 is $G(\mathcal{A}) = (A, E)$ where $A = \{1, \ldots, 8\}$, and $E = \{(1, 2), (1, 3), (3, 4), (5, 6), (5, 7), (7, 8)\}$. (This is exactly the parse forest for the set of formulas).

The treewidth of a graph or structure is a measure of how "tree-like" it is. Trees have treewidth 1, while the complete graph K_n has treewidth $n - 1$.

Definition 2. (Tree Decomposition; Treewidth).

1. *A tree decomposition of graph $G = (V, E)$ is a labelled tree $T = (U, A, L)$ with (U, A) a tree and L a function from U to 2^V such that*
   (a) *For every edge (v, w) of G, there is some $u \in U$ with $v, w \in L(u)$;*
   (b) *For each vertex v of G, the sub-graph of T induced by the set of tree vertices u with $v \in L(u)$ is connected.*
2. *The width of T is 1 less than the maximum cardinality of $L(u)$ for any $u \in U$.*

3. *The treewidth of a graph G, denoted here* $\mathrm{TW}(G)$*, is the minimum width of any tree decomposition of G.*
4. *The treewidth of a relational structure $\mathcal{A}$ is the treewidth of the Gaifman Graph of $\mathcal{A}$:* $\mathrm{TW}(\mathcal{A}) = \mathrm{TW}(G(\mathcal{A}))$.

Remark 1. Treewidth is often defined as requiring $\cup_{u \in U} L(u) = V$. It is convenient for us to omit this condition, which has no effect on treewidth because if v is independent in G we may have a distinct tree node u with $L(u) = v$.

3 Translation Schemes and Transductions

We use reductions defined by (tuples of) FO formulas. Our terminology approximately follows [13]. In defining translation schemes we allow formulas to contain a finite number of "special" constant symbols not in the vocabulary at hand, which are interpreted as themselves, and are used only for convenience in defining a domain.

Definition 3 (FO Translation Scheme). *Let τ and $\sigma = (R_1, \ldots, R_m)$ be two relational vocabularies and $C = \{c_1, \ldots\}$ a finite set of "special" constant symbols not in τ or σ. Let Φ be a tuple $\Phi = (\phi_0, \phi_1, \ldots, \phi_m)$ of $|\sigma| + 1$ FO formulas over $\tau \cup C$, where the special constants $c_i \in C$ occur only in atoms of the form $x = c_i$. Further, suppose that ϕ_0 has exactly k distinct free variables, and for each relation symbol $R_i \in \sigma$, the number of distinct free variables of the corresponding formula ϕ_i in Φ is exactly $k \cdot \mathrm{ar}(R_i)$. Then Φ is a k-ary τ-σ translation scheme.*

A τ-σ translation scheme Φ defines two functions. One is a (partial) map from τ-structures to σ-structures. This map is often called a *transduction*, generalising the use of the term in formal language theory. The second is a map from σ-formulas to τ-formulas (in model theory called an interpretation of σ in τ) that lets us answer a query about a τ-structure by translating into a query about a σ-structure. A detailed example of a translation scheme will be given in Sect. 4.

Example 4. In the usual interpretation of the complex numbers in the reals we model complex number c with real pair (r_c, i_c), and evaluate a formula over $\mathbb{C}$ by translating it into a formula over $\mathbb{R}$. The related transduction maps $\mathbb{R}$ to $\mathbb{C}$.

Definition 4 (Transduction $\overrightarrow{\Phi}$; Translation $\overleftarrow{\Phi}$). *Let $\sigma = (R_1, \ldots R_m)$ and $\Phi = (\phi_0, \phi_1, \ldots \phi_m)$ be a k-ary τ-σ translation scheme. Then:*

1. *The transduction $\overrightarrow{\Phi}$ is the partial function from τ-structures to σ-structures defined as follows. If $\mathcal{A}$ is a τ-structure and $\phi_0^{\mathcal{A}}$ is not empty, then $\mathcal{B} = \overrightarrow{\Phi}(\mathcal{A})$ is the σ-structure with*
   (a) universe $B = \{\bar{a} \in (A \cup \{c_i\})^k \mid \mathcal{A}, \bar{a} \models \phi_0\}$;
   (b) for each $i \in [1, m]$, $R_i^{\mathcal{B}} = \{\bar{a_1}, \ldots, \bar{a_i} \mid \mathcal{A}, \bar{a_1}, \ldots, \bar{a_i} \models \phi_i\}$
2. *The translation $\overleftarrow{\Phi}$ is a function from σ-formulas to τ-formulas. We obtain τ-formula $\overleftarrow{\Phi}(\psi)$ from σ-formula ψ by:*

(a) *Replacing each atom* $R_i(x_1, \ldots, x_m)$ *with* $(\wedge_j \phi_0(\bar{x_j}) \wedge \phi_i(\bar{x}_1, \ldots \bar{x}_m))$, *where each* $\bar{x_j} = (x_{j,1}, \ldots x_{j,k})$ *is a* k*-tuple of new variables;*
(b) *Replacing each existentially quantified subformula* $\exists y \psi$ *with* $\exists \bar{y}(\phi_0(\bar{y}) \wedge \psi)$, *where* $\bar{y}$ *is a* k*-tuple of new variables. Universally quantified subformulas are relativized in the dual manner.*

If $k > 1$, the universe of $\mathcal{B}$ is a set of tuples of elements of A and $\{c_i\}$, so a τ-formula that defines an r-ary relation in $\mathcal{B}$ has kr free variables. As an aid to reading, we often denote the k-tuples that make up elements of B with $\langle\rangle$.

A fundamental property of translation schemes (standard in expositions of model theory) relates their dual role defining translations and transductions.

Theorem 1 (Fundamental Property of Translation Schemes). *Let* Φ *be a* k*-ary* τ-σ *translation scheme. If* $\mathcal{A}$ *is a* τ*-structure for which* $\overrightarrow{\Phi}(\mathcal{A})$ *is defined, and* θ *is a* σ*-formula with* r *free variables* $\bar{x}$*, then*

$$\mathcal{A} \models \overleftarrow{\Phi}(\theta)(\bar{y_1}, \ldots \bar{y_r}) \Leftrightarrow \overrightarrow{\Phi}(\mathcal{A}) \models \theta(x_1, \ldots x_r)$$

where $\bar{y_i}$ *is the* k*-tuple of variables corresponding to* x_i *in the computation of* $\overrightarrow{\Phi}$.

4 FO Reductions

FO reductions are poly-time many-one reductions defined by FO transductions. Although they are weak, every problem in NP has a FO reduction to SAT.

Definition 5. *A FO reduction from a class* $\mathcal{K}$ *of* τ*-structures to a class* $\mathcal{L}$ *of* σ*-structures is a FO* τ-σ *transduction* Φ *such that* $\mathcal{A} \in \mathcal{K} \Leftrightarrow \overrightarrow{\Phi}(\mathcal{A}) \in \mathcal{L}$.

Theorem 2 ([12]). *SAT is complete for NP under FO reductions.*

To illustrate we give a translation scheme that defines a FO reduction from Propositional Satisfiability to SAT. This translation scheme may help the reader in understanding the more complex schemes described in Sects. 6 and 7.

The particular reduction is a simplified version of Tseitin's transformation from propositional formulas to CNF [20]. To transform a formula ϕ into a CNF formula of size linear in the size of ϕ, we introduce a new atom for each subformula of ϕ. Then, we write clauses over these new atoms that require the assignments made to these atoms to correspond to the values of their corresponding subformulas when ϕ is evaluated over a satisfying assignment.

Example 5. Applying the Tseitin transformation to the set of formulas $S = \{(q \wedge \neg r), (p \vee \neg t)\}$, yields the set of clauses $\{(x_1), (\neg x_1 \vee x_2), (\neg x_1 \vee x_3), (\neg x_3 \vee \neg x_4), (x_5), (\neg x_5 \vee x_6 \vee x_7), (\neg x_7 \vee \neg x_8)\}$, which is satisfiable if and only if S is. (Here, the numbering of subformulas is the same as used in Example 2).

Let $\tau_{\rm CNF} = (\mathrm{At}, \mathrm{Cl}, \mathrm{Pos}, \mathrm{Neg})$ be the vocabulary with At, Cl unary and $\mathrm{Pos}, \mathrm{Neg}$ binary. τ_{CNF}-structures represent propositional CNF formulas, with At the set of atoms, Cl the set of clauses, and Pos(a, c) (resp. Neg(a, c)) meaning that atom a occurs positively (resp., negatively) in clause c.

Let $\Phi_T = (\phi, \phi_{\rm At}, \phi_{Cl}, \phi_{\rm Pos}, \phi_{\rm Neg})$, be the translation scheme defined by the following formulas, with special constants {atom,topClause,orClause, andClause, notClause}.

(i) $\phi(\langle s,i\rangle) = [(\mathrm{SubF}(i) \wedge (s = \mathrm{atom})) \vee (\exists j \mathrm{Or}(j,i) \wedge (s = \mathrm{orClause})) \vee (\exists j \mathrm{And}(j,i) \wedge (s = \mathrm{andClause})) \vee (\mathrm{Form}(i) \wedge (s = \mathrm{topClause})) \vee (\mathrm{Form}(i) \wedge (s = \mathrm{notClause}))]$

(ii) $\phi_{At}(s,i) = [\mathrm{SubF}(i) \wedge (s = \mathrm{atom})]$

(iii) $\phi_{Cl}(s,i) = [(\exists x \mathrm{Or}(i,x) \wedge (s = \mathrm{orClause})) \vee (\exists x \mathrm{And}(x,i) \wedge (s = \mathrm{andClause})) \vee (\exists x \mathrm{Not}(i,x) \wedge (s = \mathrm{notClause})) \vee (\mathrm{Form}(i) \wedge (s = \mathrm{topClause}))]$

(iv) $\phi_{Neg}(\langle s,i\rangle, \langle c,j\rangle) = ((s = \mathrm{atom}) \wedge [(\exists x \mathrm{Or}(i,x) \wedge (c = \mathrm{orClause}) \wedge j = i) \vee (\mathrm{And}(i,j) \wedge (c = \mathrm{andClause})) \vee (\mathrm{Not}(j,i) \wedge (c = \mathrm{notClause})) \vee (\exists x \mathrm{Not}(x,i) \wedge j = i \wedge (c = \mathrm{notClause}))])$

(v) $\phi_{Pos}(\langle s,i\rangle, \langle c,j\rangle) = ((s = \mathrm{atom}) \wedge [(\exists x \mathrm{And}(x,i) \wedge j = i \wedge (c = \mathrm{andClause})) \vee (\mathrm{Or}(i,j) \wedge (c = \mathrm{orClause})) \vee (\mathrm{Form}(i) \wedge i = j \wedge (c = \mathrm{topClause}))])$

Then Φ_T defines a FO transduction that carries out Tseitin's transformation of propositional formulas to CNF formulas. So, $\overrightarrow{\Phi_T}$ is a FO reduction from Propositional Satisfiability to SAT.

Let $\mathcal{A}$ be the τ_{PL}-structure representing a propositional formula ϕ, let $\mathcal{B}$ be the structure $\mathcal{B} = \Phi_T(\mathcal{A})$, and Γ be the CNF formula represented by the τ_{CNF}-structure $\mathcal{B}$. The domain B of $\mathcal{B}$ contains an element $\langle atom, i\rangle$ for each subformula i of ϕ, and these correspond exactly to the atoms in formula Γ. B also contains an element $\langle x, i\rangle$ for each clause in Γ. In each of these elements, the x identifies the role of the clause, and the i identifies the subformula of ϕ that it corresponds to. For example, the domain element $\langle orClause, 3\rangle$ would be associated with subformula 3 being a disjunction. The roles of clauses are: topClause, notClause, andClause, orClause. In the reduction, we have a single clause for each disjunction, a single clause for each negation, and two clauses for each conjunction in ϕ. The correspondence we use associates the clause for a disjunction or negation with the corresponding subformula, but we associate the two clauses for a conjunction with the two conjuncts. There is also an element for the top clause $\langle \mathrm{topClause}, i\rangle$ of each formula i.

Example 6. If $\mathcal{A}$ is the structure of Example 2 then the structure $\mathcal{B} = \overrightarrow{\Phi_T}(\mathcal{A})$ is, in part, as follows. Domain $B = At^{\mathcal{B}} \cup Cl^{\mathcal{B}}$; $At^{\mathcal{B}} = \{\langle \mathrm{atom}, 1\rangle, \langle \mathrm{atom}, 2\rangle, \ldots, \langle \mathrm{atom}, 8\rangle\}$; $Cl^{\mathcal{B}} = \{\langle \mathrm{topClause}, 1\rangle, \langle \mathrm{topClause}, 5\rangle, \langle \mathrm{andClause}, 2\rangle, \langle \mathrm{andClause}, 3\rangle, \langle \mathrm{orClause}, 5\rangle, \langle \mathrm{notClause}, 3\rangle, \langle \mathrm{notClause}, 7\rangle\}$; $Pos^{\mathcal{B}} = \{\langle\langle \mathrm{atom}, 1\rangle, \langle \mathrm{topClause}, 1\rangle\rangle, \langle\langle \mathrm{atom}, 5\rangle, \langle \mathrm{topClause}, 5\rangle\rangle, \ldots\}$; ... This represents the CNF formula: $\{(\langle \mathrm{atom}, 1\rangle), (\langle \mathrm{atom}, 5\rangle), (\neg\langle \mathrm{atom}, 1\rangle, \langle \mathrm{atom}, 2\rangle), (\neg\langle \mathrm{atom}, 1\rangle, \langle \mathrm{atom}, 3\rangle), \ldots\}$. Under the map $\langle atom, i\rangle \mapsto x_i$, this is the formula obtained in Example 5.

The well-known meta-theorem tells us that many problems are fixed-parameter tractable for treewidth. MSO is the fragment of second order logic in which second order variables must be monadic.

Theorem 3 (Courcelle [4]). *Every MSO-definable class of structures can be recognized by an algorithm that runs in time $f(w)O(n)$, where n is the size of the encoding of the structure and w the treewidth of the structure, and f is a computable function.*

Example 7. It is straightforward to write an MSO formula, in the vocabulary τ_{CNF}, defining SAT: $\exists S[\forall x(S(x) \rightarrow At(x)) \wedge (\forall y(Cl(y) \rightarrow \exists z((Pos(z,y) \wedge S(z)) \vee (Neg(z,y) \wedge \neg S(z)))))]$. Here, the monadic second order variable S is the set of atoms made true by a satisfying assignment.

Example 8. We can define formula satisfiability with an MSO formula over vocabulary τ_{PL}. Such a formula can be obtained from the formula of Example 7 and the translation scheme Φ_T using Theorem 1.

5 Guarded Reductions

In this section, we define a family of guarded reductions and show that these reductions are treewidth preserving.

Definition 6 (Treewidth Preserving Reduction). *We say a reduction f from $\mathcal{L}$ to $\mathcal{K}$ is bounded treewidth preserving (or just treewidth preserving) if there is a computable function g such that, for every $\mathcal{A} \in \mathcal{L}$,* $\mathrm{TW}(f(\mathcal{A})) \leq g(\mathrm{TW}(\mathcal{A}))$.

5.1 Treewidth of CNF Formulas

Treewidth of CNF formulas is usually defined in terms of on one of two graphs associated CNF formula. The primal graph has a vertex for each atom, and an edge (u, v) iff u and v occur together in a clause (regardless of polarity). The incidence graph has a vertex for each atom and for each clause, and an edge (a, c) if atom a occurs in clause c (with either polarity). These induce two notions of treewidth for CNF formulas, the primal treewidth and incidence treewidth. Since the indcidence treewidth is at most one more than the primal treewidth (and sometimes much smaller), it is the more interesting measure.

For every propositional CNF formula Γ, the Gaifman graph of the τ_{CNF}-structure for Γ is the incidence graph of Γ. Therefore, in this paper the treewidth of a CNF formula means its incidence treewidth. (In some work the primal graph is called the Gaifman graph. This results from a different association of structures with CNF formulas).

An example of a treewidth preserving reduction to SAT is the usual reduction from 3-Colouring. Each vertex is mapped to 3 atoms (one for each colour). For each vertex there is a clause saying it must be coloured, and for each edge three clauses say the ends have distinct colours. A graph of treewidth w is mapped to a CNF formula of treewidth $3w$.

If P ≠ NP there are problems in NP which do not have treewidth preserving reductions to SAT. SAT is FPT for treewidth, so any problem with a treewidth preserving reduction to SAT must also be FPT for treewidth. However, there are problems that are NP-complete on trees (e.g. Call Scheduling [6] and Common Embedded Subtree [10]) or on bounded treewidth graphs (e.g. Edge Disjoint Paths [17], and Weighted Colouring [15]), and thus not FPT for treewidth.

5.2 Guarded FO Reductions

The primitive positive formulas are the smallest set of FO formulas containing all atoms, including those of the form $x = y$, and closed under conjunction and existential quantification.

Definition 7. *Let $\phi(\bar{x})$ be a FO formula. A FO formula γ is a* packed guard *for ϕ if it is a primitive positive formula of the form $\gamma_1 \wedge \ldots \wedge \gamma_m$, where each γ_i is either an atom or an existentially quantified atom, such that:*

1. *free(γ) = free(ϕ) = $\bar{x}$;*
2. *Each pair y, z of distinct variables from $\bar{x}$ appears among the free variables of some γ_i in γ.*

The name is taken from the Packed Fragment of FO, introduced in [14], in which guards are of this form.

Definition 8 (Guarded Translation Scheme, Guarded Reduction). *A FO τ-σ translation scheme Φ is guarded if every formula in the scheme, except possibly the domain-defining formula ϕ_0, is a disjunction of formulas of the form $(\gamma(\bar{x}) \wedge \psi(\bar{x}))$, where γ is a packed guard for ψ. A guarded reduction is a reduction defined by a guarded translation scheme.*

The guards relativize quantification to the contents of instance relations. They ensure that, if Φ is a guarded translation scheme, every edge of the Gaifman graph of $\mathcal{B} = \overrightarrow{\Phi}(\mathcal{A})$ has a corresponding edge in the Gaifman graph of $\mathcal{A}$. To see this, consider a tuple $\bar{b}$ in a relation of $\mathcal{B}$. If $\bar{b}$ contributes an edge to the Gaifman graph, it contains at least two elements. These elements are constructed from tuples of elements from A. Any two of these elements had to co-occur in the relation defined by one of the atoms of a guard formula, and therefore has a corresponding edge in the Gaifman graph of $\mathcal{A}$.

5.3 Guarded Reductions Preserve Bounded Treewidth

To show guarded reductions preserve bounded treewidth we construct a small-width tree decomposition of $\overrightarrow{\Phi}(\mathcal{A})$ from a small-width decomposition of $\mathcal{A}$.

Theorem 4. *Let Φ be a guarded k-ary τ-σ translation scheme. If $\mathcal{A}$ is a τ-structure with $\mathrm{TW}(\mathcal{A}) \leq w$ and $\mathcal{B} = \overrightarrow{\Phi}(\mathcal{A})$ then $\mathrm{TW}(\mathcal{B}) \leq (w+1)^k$.*

Proof. Let $\mathcal{B} = \overrightarrow{\Phi}(\mathcal{A})$, and let $T_{\mathcal{A}} = (U, F, L_{\mathcal{A}})$ be a width w tree decomposition of $\mathcal{A}$. We will construct a tree decomposition $T_{\mathcal{B}} = (U, F, L_{\mathcal{B}})$ of $\mathcal{B}$ that is isomorphic to $T_{\mathcal{A}}$, but with a different labelling function (a.k.a. "bag" contents). Let $\mathbf{B}$ be the set of all tuples occurring in some relation of $\mathcal{B}$. Construct a total function $f : \mathbf{B} \to U$ as follows. Let $\bar{b} = (b_1, \ldots, b_r)$ be a tuple in a relation of $\mathcal{B}$. Then $\bar{b}$ is of the form $(\langle a_{1,1}, \ldots a_{1,k}\rangle, \ldots \langle a_{r,1}, \ldots a_{r,k}\rangle)$. By construction of the guards of Φ, each pair $(a_{i,j}, a_{i',j'})$ in $\bar{b}$ has a corresponding edge in the Gaifman graph of $\mathcal{A}$. Therefore, there is a clique in $G(\mathcal{A})$ containing all of the elements $a_{i,j}$ in $\bar{b}$. It follows that there must be a vertex $u \in U$ of $T_{\mathcal{A}}$ such that $\{\, a \mid a \text{ occurs in } \bar{b}\} \subset L_{\mathcal{A}}(u)$. Let $f(\bar{b})$ be such a u. Now define $L_{\mathcal{B}}$ in terms of f by $L_{\mathcal{B}}(u) = \cup\{\, \bar{b} \mid \bar{b} \in \mathbf{B} \text{ and } f(\bar{b}) = u\}$. The first condition for $T_{\mathcal{B}}$ to be a tree decomposition of $\mathcal{B}$ is now satisfied. We establish the second condition by modifying $T_{\mathcal{B}}$ as follows: If for some $b \in B$ and for two distinct tree nodes u,v we have that $b \in L_{\mathcal{B}}(u)$ and $b \in L_{\mathcal{B}}(v)$ but $b \notin L_{\mathcal{B}}(w)$, we add b to $L_{\mathcal{B}}(w)$. It remains to establish an upper bound on the size of bags (the sets $L_{\mathcal{B}}(u)$). Each element of $b \in B$ is a tuple $b = \langle a_1, \ldots a_k\rangle$ of k elements from A. By construction of $T_{\mathcal{B}}$, if $b \in L_{\mathcal{B}}(u)$ then for each $a_i \in b$, we have that $a_i \in L_{\mathcal{A}}(u)$. The bound is established by the number of elements in $L_{\mathcal{A}}(u)$ and the number of elements $b \in B$ that could be constructed from these. This number is at most $(w+1)^k$, since an element of B is a k-tuple of elements from $L_A(u)$. □

Preservation of bounded treewidth, follows immediately.

Theorem 5. *Let Γ be a guarded reduction from $\mathcal{K}$ to $\mathcal{L}$. Then there is a computable function f such that, for every $\mathcal{A} \in \mathcal{K}$*

$$\mathrm{TW}(\Gamma(\mathcal{A})) \leq f(\mathrm{TW}(\mathcal{A}))$$

6 Automatically Generating Reductions

We now consider how specifications induce reductions. We consider a problem specification to be a formula Ψ of the form $\exists \bar{R}\psi$, where $\bar{R}$ is a tuple of second order variables, and ψ is a FO sentence. If the problem defined is a class of τ-structures, then Ψ is a τ-formula, and ψ is a formula with vocabulary $(\tau, \bar{R})$. The decision problem is: given a τ-structure $\mathcal{A}$, decide if $\mathcal{A} \models \Psi$. The associated search problem is to find a witness for the existential SO variables, or, equivalently, to find a $\tau \cup \bar{R}$-structure $\mathcal{B} = (\mathcal{A}, \bar{\mathcal{R}}^{\mathcal{B}})$ that is an expansion of $\mathcal{A}$ to the vocabulary of ψ, and such that $\mathcal{B} \models \psi$.

We may regard $\Psi = \exists \bar{R}\psi$ as implicitly defining a reduction to SAT, based on the following four-step construction:

1. Given $\mathcal{A}$, construct a quantifier-free formula from ψ by rewriting each quantified subformula as a large conjunction or disjunction over elements of A;
2. Transform the resulting ground formula to CNF by Tseitin's method;
3. Evaluate any atoms over the vocabulary τ of $\mathcal{A}$, deleting any clauses that evaluate to true;
4. Replace each distinct ground FO atom with a distinct propositional atom.

For Step 1, we must introduce new constant symbols to denote elements of A in the ground formula. For Step 2, we must introduce new atoms corresponding to subformulas, to be used in the Tseitin construction. For the purpose of associating tree decompositions of the final propositional CNF formula with tree decompositions of $\mathcal{A}$, we will construct these "Tseitin" atoms as ground FO atoms using new relation symbols. Roughly speaking, the result is a ground formula Γ such that models of Γ correspond to the expansions of $\mathcal{A}$ that witness the SO existentials in Ψ.

We write $\tilde{A}$ for the set of (new) constant symbols $\tilde{A} = \{\tilde{a} | a \in A\}$. Then, the first step of our construction is defined by the following recursive function $\Gamma(\phi, \nu, A)$. Here ϕ is a FO formula, ν is a partial map from variables to domain elements, $\nu\langle x \to a\rangle$ is the valuation just like ν except that it maps x to a, and A is the domain.

$$\Gamma(\phi,\nu,A) = \begin{cases} \phi(\nu(\bar{x})) & \text{if } \phi \text{ is an atom } \phi(\bar{x}) \\ (\Gamma(\psi_1,\nu,A) \vee \Gamma(\psi_2,\nu,A)) & \text{if } \phi \text{ is } (\psi_1 \vee \psi_2) \\ (\Gamma(\psi_1,\nu,A) \wedge \Gamma(\psi_2,\nu,A)) & \text{if } \phi \text{ is } (\psi_1 \wedge \psi_2) \\ \neg\Gamma(\psi,\nu,A) & \text{if } \phi \text{ is } \neg\psi \\ (\bigwedge_{a\in A} \Gamma(\psi, \nu\langle x \to \tilde{a}\rangle,\ A)) & \text{if } \phi \text{ is } \forall x\psi \\ (\bigvee_{a\in A} \Gamma(\psi, \nu\langle x \to \tilde{a}\rangle,\ A)) & \text{if } \phi \text{ is } \exists x\psi \end{cases}$$

For step 2, associate with each non-atomic subformula η of ψ a new relation symbol P_η, with arity $|\text{free}(\eta)|$. Associate with each non-atomic subformula β of $\Gamma(\psi, A) = \Gamma(\psi, \emptyset, A)$ a ground atom $P\bar{a}$, where P is a relation symbol associated to the corresponding subformula of ψ, and $\bar{a} = \nu(\text{free}(\psi))$ with ν the substitution used in evaluating $\Gamma(\psi, \nu, A)$ in constructing $\Gamma(\psi, A)$. Now, apply Tseitin's reduction to CNF to the formula $\Gamma(\psi, A)$, using the atoms just defined as the Tseitin atoms corresponding to the non-atomic subformulas. Denote the resulting formula $\beta = \text{CNF}(\Gamma(\psi, A))$.

β is a ground FO formula in CNF form, over atoms with constant symbols from $\tilde{A}$, with each relation symbol either a vocabulary symbol of Ψ, a second order variable symbol of Ψ, or a Tseitin symbol as just introduced. Step 3 is to eliminate atoms over τ by replacing them with their truth values determined by $\mathcal{A}$. More precisely, we delete each clause that contains a true atom and delete all false atoms from remaining clauses.

Definition 9 (Grounding). *Let Ψ be a $\exists$SO τ-formula $\exists\bar{R}\psi$ where ψ is a FO sentence. Let $\mathcal{A}$ be a τ-structure. We call a formula Γ a grounding of Ψ over $\mathcal{A}$ if it satisfies the following properties:*

1. *Γ is a ground formula for a vocabulary σ that includes $\tau, \bar{R}, \tilde{A}$;*
2. *If $\mathcal{A} \models \Psi$ then there is an expansion $\mathcal{B}$ of $\tilde{\mathcal{A}}$ to σ such that $\mathcal{B} \models \Gamma$;*
3. *If $\mathcal{B}$ is a σ-structure that is an expansion of $(A, \tilde{A})$ and $\mathcal{B} \models \Gamma$ and $\mathcal{C}$ is the τ-reduct of $\mathcal{B}$, then $\mathcal{C} \models \Psi$.*

So $\Gamma = \text{CNF}(\Gamma(\psi, A))$ is a grounding. Of particular interest are certain proper subsets of $\text{CNF}(\Gamma(\psi, A))$ that also are groundings.

Definition 10 (Direct Grounding). *We call a formula Ψ a direct grounding of ψ over $\mathcal{A}$ if it satisfies the following:*

1. *Ψ is a grounding of ψ over $\mathcal{A}$*
2. *For every clause C of Ψ, there is a clause C' of $\mathrm{CNF}(\Gamma(\psi, A))$ with $C \subseteq C'$.*

Any subformula of ψ that contains no symbols from $\bar{R}$ can be directly evaluated over $\mathcal{A}$. This can be used to reduce the number of clauses included in a grounding of Ψ over $\mathcal{A}$. In particular, consider subformula $\psi(\bar{x})$ of ψ of the form $\psi_1(\bar{x}) \wedge \psi_2(\bar{x})$, and suppose that ψ_2 contains symbols from $\bar{R}$ but ψ_1 does not. Then, for each substitution ν for which $\psi_1(\nu(\bar{x}))$ evaluates to false, the clauses corresponding to $\Gamma(\psi_2, \nu, A)$ may be left out of Γ, and it will still be a direct grounding. A dual property holds for disjunctive subformulas. We say that a grounding that leaves out such clauses satisfies the *lazy generation property*. Practical grounding software has this property. In a direct recursive implementation of $\Gamma(\phi, \nu, A)$, lazy generation amounts to little more than lazy evaluation.

6.1 Direct CNF Grounding as a FO Transduction

We wish to show that, from a $\exists$SO problem specification $\Psi = \exists \bar{R}\psi$, we can (algorithmically) construct a FO reduction Δ from the class of models of Ψ to SAT. The image of $\mathcal{A}$ under Δ is a structure for vocabulary $\tau_{CNF} = (\mathrm{At}, \mathrm{Cl}, \mathrm{Pos}, \mathrm{Neg})$. So, the formulas of Δ will have certain elements in common with those of our transduction Φ_T from Sect. 4.

As before, the domain B of $\mathcal{B} = \Delta(\mathcal{A})$ has elements corresponding to atoms and clauses of the resultant CNF formula. It needs an element identified with each ground atom $P\bar{a}$, where P is an element of $\bar{R}$ or a Tseitin relation symbol corresponding to a subformula of Ψ, and $\bar{a}$ is a tuple of elements of A. Let r be the maximum number of free variables in a subformula of Ψ. Then our domain elements will be $k = r + 2$-tuples $\langle P, \bar{a}, C\rangle$, where P is a special constant symbol denoting a relation symbol, C is a special constant symbol denoting an atom or clause-type from {atom,topClause, orClause,andClause,notClause}, and $\bar{a}$ is a tuple from $(A \cup \{\_\})^w$. The special constant $\_$ is a place-holder letting us model a tuple $\bar{a}$ of arity less than w with a w-tuple.

For each subformula of ψ that is a disjunction, our ground CNF formula has one ternary clause for every instantiation of the free variables. For example, the disjunction $\phi(\bar{x}) = (\phi_1(\bar{x}) \vee \phi_2(\bar{x}))$ contributes a clause of the form $(\neg P_\phi \bar{a} \vee P_{\phi_1}\bar{a} \vee P_{\phi_2}\bar{a})$ for each instantiation $\bar{a}$ of its free variables. Such a clause contributes three pairs to relations of $\mathcal{B}$: one in $Neg^{\mathcal{B}}$ and two in $Pos^{\mathcal{B}}$, as in the propositional case. Supposing $\bar{x}$ to be of size 2, the two in Pos, for all instantiations, can be defined by a formula

$$\begin{aligned}\alpha(\langle p, x_1, \ldots, x_k, c_1\rangle, \langle p, x_1, \ldots, x_k, c_2\rangle) = \\ (p = P_\phi \ \wedge (x_1 = x_1) \wedge (x_2 = x_2) \wedge (x_3 = \_) \wedge \ldots \wedge (x_k = \_) \\ \wedge (c_1 = \mathrm{atom}) \wedge (c_2 = \mathrm{orClause}))\end{aligned} \quad (1)$$

Similarly, for each subformula of the form $\exists x\phi$, we have $|A|+1$ pairs in $\mathcal{B}$, one in $Neg^{\mathcal{B}}$ and the rest in $Pos^{\mathcal{B}}$. We proceed similarly for all connectives and for the atoms. The formula ϕ_P of Δ, that defines the relation $Pos^{\mathcal{B}}$ is defined by the disjunction of all formulas defining particular subsets of $Pos^{\mathcal{B}}$, and similarly for $Neg^{\mathcal{B}}$. From the complete construction, we obtain the following.

Theorem 6. *For every $\exists$SO formula Ψ, there is a FO transduction Δ that is a reduction from Mod Ψ to SAT. In particular, $\Delta(\mathcal{A})$ is a direct grounding of Ψ over $\mathcal{A}$.*

7 Guarded Existential SO Specifications

As in other guarded logics, we assume FO quantifiers apply to blocks of variables, and that every formula of the form $\exists x \exists y \phi$ has been re-written as $\exists xy\, \phi$, and similarly for $\forall$.

Definition 11 (Guarded $\exists$SO). *Call an $\exists$SO formula $\Psi = \exists \bar{R}\psi$ guarded if*

1. *In every subformula that is of the form $\exists \bar{x}\phi$ and that contains a non-monadic symbol from $\bar{R}$, ϕ is of the form $(\gamma(\bar{y}) \wedge \phi'(\bar{y}))$, where γ is a packed guard for ϕ', and $\bar{y} \supset \bar{x}$.*
2. *In every subformula that is of the form $\forall \bar{x}\phi$ and that contains a non-monadic symbol from $\bar{R}$, ϕ is of the form $(\gamma(\bar{y}) \rightarrow \phi'(\bar{y}))$, where γ is a packed guard for ϕ', and $\bar{y} \supset \bar{x}$.*

In practice, guarded specifications are very natural. For example, consider the Vertex Cover problem, in which we are given a graph $G = (V, E)$ and must find a set S (normally with some size bound) containing at least one end point of each edge. This property is naturally described with the guarded formula

$$\forall u, v(Euv \rightarrow (Su \vee Sv)).$$

However, for the Domatic Partition problem, which calls for a partition of vertices into sets $P_1, \ldots, P_k$, it does not seem that there is a guarded version (the only possible guard being E) of the property:

$$\forall v \exists i(Pvi \wedge \forall j(Pvj \rightarrow j = i)).$$

Theorem 7. *Let Ψ be a guarded $\exists$SO formula with vocabulary τ, and Δ a reduction that implements a direct grounding of Ψ over $\mathcal{A}$ that satisfies the lazy evaluation property. Then for any τ-structure $\mathcal{A}$, we have that*

$$TW(\Delta(\mathcal{A})) \leq f(TW(\mathcal{A}))$$

where $f(x) = O((x+1)^r)$, with r determined by Ψ.

The proof is quite similar to that of Theorem 4.

Proof (Sketch). Let T be $T_{\mathcal{A}} = (U, F, L_{\mathcal{A}})$ be a width w tree decomposition of $\mathcal{A}$, and Γ be the formula $\Gamma = CNF(\Gamma(\Psi, \mathcal{A}))$. We will construct a tree decomposition $T_{\mathcal{B}}$ for Γ that is isomorphic to $T_{\mathcal{A}}$, but has different bags. Consider any existentially quantified subformula $\psi'(\bar{y})$ of Ψ that has a symbol from $\bar{R}$. This formula is of the form $\exists x(\gamma(\bar{y}, x) \wedge \psi(\bar{y}, x))$, where γ is a packed guard for ψ. By the lazy generation property, no clauses corresponding to $\gamma(\nu(\bar{y}, x))$ are included in Γ, and for each tuple $\bar{a} = \nu(\bar{y}, x)$, clauses corresponding to $\psi(\nu(\bar{y}, x))$ are only included in Γ if $\Gamma(\nu(\bar{y}, x))$ evaluates to true. Consider the formula $\psi(\nu(\bar{y}, x))$. If it is quantifier-free, then all atoms in corresponding clauses of Γ are of the form $P\bar{a}$ where P is either in $\bar{R}$ or a Tseitin symbol corresponding to a subformula of ψ. Since $\bar{a}$ was "sanctioned" by the guard, we know that there is a bag in $T_{\mathcal{A}}$ containing all elements of $\bar{a}$. We put all the atoms from all the clauses into the corresponding bag in $T_{\mathcal{B}}$. If ψ has quantified subformulas, then we include as part of the current step the Tseitin atoms corresponding to those subformulas, but not the clauses corresponding to them. Following this, we have that the first of the tree decomposition properties is satisfied by $T_{\mathcal{B}}$. As in the proof of Theorem 4, we add to each bag the minimum collection of atoms to satisfy the second property. It then remains to bound the size of the bags. As in the proof of Theorem 4, it is sufficient to bound the number of elements in a bag of $T_{\mathcal{B}}$ in terms of the number in a related bag of $T_{\mathcal{A}}$. From $w + 1$ elements of A, in a bag $L_A(u)$ we can construct $\binom{w+1}{r}$ tuples. The number of relation symbols is bounded by $|\Psi|$, and there are 5 special constants (which were used to distinguish the syntactic types of subformulas) for the last element, so the number of elements in a $L_B(u)$ is at most

$$5|\Psi|\binom{w+1}{r} = O((w+1)^r)$$

which is polynomial in w because the constant 5, $|\Psi|$ and r are all fixed by Ψ. □

7.1 Guarded FO Reductions from Guarded Specifications

To obtain a guarded FO reduction from a guarded ∃SO specification, we first construct an FO reduction according to the process in Sect. 6.1. We then modify it by conjoining a suitable guard to each disjunct of each formula defining the reduction. The appropriate guard for the formula defining elements of $N^{\mathcal{B}}$ or $P^{\mathcal{B}}$ corresponding to a subformula ϕ of the specification, is the guard for the least subformula of the specification ψ that is quantified and that contains ϕ as a subformula. Consider the formula 1 of Sect. 6.1. Suppose that the subformula of Ψ it addresses, $(\phi_1(x_1, x_2) \vee \phi_2(x_1, x_2))$, appears in a subformula of the form

$$\exists x_1((\gamma(x_1, x_2) \wedge (\phi_3(x_1) \wedge ((\phi_1(x_1, x_2) \vee \phi_2(x_1, x_2))))).$$

Then we add the guard $\gamma(x_1, x_2)$, to obtain the guarded formula

$$\begin{aligned}\alpha(\langle p, x_1, \ldots, x_k, c_1\rangle, \langle p, x_1, \ldots, x_k, c_2\rangle) = [\gamma(x_1, x_2) \wedge \\ (p = P_\phi \ \wedge (x_1 = x_1) \wedge (x_2 = x_2) \wedge (x_3 = \_) \wedge \ldots \wedge (x_k = \_) \\ \wedge (c_1 = \text{atom}) \wedge (c_2 = \text{orClause}))] \qquad (2)\end{aligned}$$

From the complete construction, we obtain the following.

Theorem 8. *For every guarded ∃SO formula Ψ, there is a guarded FO reduction Δ from Mod Ψ to SAT.*

8 Discussion

When writing specifications in constraint modelling or knowledge representation languages, it is common practice to write constraints in guarded form when this is easy. Our results demonstrate one possible benefit of this, and also suggest that making an effort to write guarded specifications might improve solving time. The potential speedup does not depend on instances being of bounded treewidth, since guarded reductions will preserve related sparseness properties for instance families that are somehowe "close to" having small treewidth. More importantly, our work takes a step toward understanding when reductions obtained from declarative problem specifications may preserve interesting structural instance properties.

We would like to also obtain necessary conditions for existence of treewidth preserving reductions. We conjecture that the class of problems with treewidth preserving reductions to SAT is strictly larger than the class of problems with guarded FO reductions to SAT. We also conjecture that guarded reductions preserve structural sparseness properties more general than bounded treewidth.

Related Work. Bliem et al. [2] demonstrated treewidth affecting ASP solver run-time, and introduced connection guarded ASP programs. These preserve bounded treewidth in grounding, but only if degree is also bounded. Bliem [1] defined guarded ASP programs and showed that grounding for these preserves bounded treewidth regardless of degree. The definition is quite restrictive, and we conjecture there are problems with no guarded ASP formulas but with guarded ∃SO definitions and guarded reductions to SAT. However, it is possible that with a carefully formulated use of defined predicates in guards this could be remedied. The paper [3] illustrates extending the features of the IDP system by using the system itself to compute transductions. Results in [8,18] indicate that there should be very efficient algorithms for grounding guarded specifications. The MSO transductions in [4] preserve treewidth but are restricted in a way that makes them too weak for our application.

Acknowledgements. Phokion Kolaitis suggested studying "good" reductions by via special classes such as FO reductions [11]. Marc Denecker suggested that guarded formulas should produce groundings with bounded treewidth [5]. This work was supported in part by an NSERC Discovery Grant.

References

1. Bliem, B.: ASP programs with groundings of small treewidth. In: Ferrarotti, F., Woltran, S. (eds.) FoIKS 2018. LNCS, vol. 10833, pp. 97–113. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90050-6_6

2. Bliem, B., Moldovan, M., Morak, M., Woltran, S.: The impact of treewidth on ASP grounding and solving. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 852–858 (2017)
3. Bogaerts, B., Jansen, J., de Cat, B., Janssens, G., Bruynooghe, M., Denecker, M.: Bootstrapping inference in the IDP knowledge base system. New Gener. Comput. **34**(3), 193–220 (2016)
4. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic -A Language-Theoretic Approach. Encyclopedia of Mathematics and Its Applications, vol. 138. Cambridge University Press, Cambridge (2012)
5. Denecker, M.: Personal communication (2015)
6. Erlebach, T., Jansen, K.: Call scheduling in trees, rings and meshes. In: 30th Annual Hawaii International Conference on System Sciences (HICSS-30), 7–10 January 1997, Maui, Hawaii, USA, p. 221 (1997)
7. Fagin, R.: Generalized first-order spectra and polynomial-time recognizablesets'. In: Proceedings of the SIAM-AMS, vol. 7 (1974)
8. Flum, J., Frick, M., Grohe, M.: Query evaluation via tree-decompositions. J. ACM **49**(6), 716–752 (2002)
9. Freuder, E.C.: A sufficient condition for backtrack-free search. J. ACM **29**(1), 24–32 (1982)
10. Kilpelainen, P., Mannila, H.: Ordered and unordered tree inclusion. SIAM J. Comput. **24**(2), 340–356 (1995)
11. Kolaitis, P.: Personal communication (2014)
12. Lovász, L., Gács, P.: Some remarks on generalized spectra. Math. Log. Q. **23**(36), 547–554 (1977)
13. Makowsky, J.A.: Algorithmic uses of the Feferman-Vaught theorem. Ann. Pure Appl. Log. **126**(1–3), 159–213 (2004)
14. Marx, M.: Tolerance logic. J. Log. Lang. Inform. **10**(3), 353–374 (2001)
15. McDiarmid, C., Reed, B.: Channel assignment on graphs of bounded treewidth. Discrete Math. **273**(1), 183–192 (2003). EuroComb 2001
16. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74970-7_38
17. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is NP-complete for series-parallel graphs. Discrete Appl. Math. **115**(1), 177–186 (2001). First Japanese-Hungarian Symposium for Discrete Mathematics and its Applications
18. Patterson, M., Liu, Y., Ternovska, E., Gupta, A.: Grounding for model expansion in k-guarded formulas with inductive definitions. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, 6–12 January 2007, pp. 161–166 (2007)
19. Samer, M., Szeider, S.: Constraint satisfaction with bounded treewidth revisited. J. Comput. Syst. Sci. **76**(2), 103–114 (2010)
20. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Siekmann, J.H., Wrightson, G. (eds.) Automation of Reasoning. Symbolic Computation (Artificial Intelligence), pp. 466–483. Springer, Heidelberg (1983). https://doi.org/10.1007/978-3-642-81955-1_28