Constructing NP-intermediate Problems by Blowing Holes with Parameters of Various Properties[☆]

Peter Jonsson[a], Victor Lagerkvist[a,*], Gustav Nordh[b]

[a]*Department of Computer and Information Science, Linköpings Universitet, Sweden.*
[b]*Kvarnvägen 6, 53374, Hällekis, Sweden.*

Abstract

The search for natural NP-intermediate problems is one of the holy grails within computational complexity. Ladner's original diagonalization technique for generating NP-intermediate problems, blowing holes, has a serious shortcoming: it creates problems with a highly artificial structure by arbitrarily removing certain problem instances. In this article we limit this problem by generalizing Ladner's method to use parameters with various characteristics. This allows one to define more fine-grained parameters, resulting in NP-intermediate problems where we only blow holes in a controlled subset of the problem. We begin by fully characterizing the problems that admit NP-intermediate subproblems for a broad and natural class of parameterizations, and extend the result further such that structural CSP restrictions based on parameters that are hard to compute (such as tree-width) are covered, thereby generalizing a result by Grohe. For studying certain classes of problems, including CSPs parameterized by constraint languages, we consider more powerful parameterizations. First, we identify a new method for obtaining constraint languages Γ such that CSP(Γ) are NP-intermediate. The sets Γ can have very different properties compared to previous constructions (by, for instance, Bodirsky & Grohe) and provides insights into the algebraic approach for studying the complexity of infinite-domain CSPs. Second, we prove that the propositional abduction problem parameterized by constraint languages admits NP-intermediate problems. This settles an open question posed by Nordh & Zanuttini.

Keywords: Computational complexity, NP-intermediate problems, Constraint satisfaction problems, Abduction problems

[☆]A preliminary version of this article appeared in *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming (CP-2013), pp. 398-414. Uppsala, Sweden, Sep, 2013.*

[*]Corresponding author.

Email addresses: `peter.jonsson@liu.se` (Peter Jonsson), `victor.lagerkvist@liu.se` (Victor Lagerkvist), `gustav.nordh@gmail.com` (Gustav Nordh)

1. Introduction

Background

Assuming P $\neq$ NP it is natural to consider problems in NP $\setminus$ P which are not NP-hard. Such problems are referred to as *NP-intermediate*, and Ladner [30] explicitly constructed NP-intermediate problems by removing strings of certain lengths from NP-complete languages via a diagonalization technique that is colloquially known as *blowing holes in problems.* The languages constructed via blowing are unfortunately famous for being highly artificial: Arora and Barak [3] write the following.

> We do not know of a natural decision problem that, assuming NP $\neq$ P, is proven to be in NP $\setminus$ P but not NP-complete, and there are remarkably few candidates for such languages.

More natural examples are known under other complexity-theoretic assumptions. For instance, LogClique (the problem of deciding whether an n-vertex graph contains a clique of size $\log n$) is NP-intermediate under the exponential-time hypothesis (ETH). We wish to stress the difference between problems that, assuming P $\neq$ NP, are provably neither P nor NP-complete, and problems whose complexity is simply undetermined at the moment. As for the latter, of the dozen problems in Garey & Johnson [20] which at the time where not known to be P or NP-complete, only a few, such as the integer factorization problem and the graph isomorphism problem, remain unresolved. The integer factorization problem is particularly interesting in this sense: it is not likely to be NP-complete since it is both in NP and coNP, and no polynomial-time algorithm is known despite considerable efforts to construct one. The lack of natural NP-intermediate computational problems makes it important to investigate new classes of NP-intermediate problems and, hopefully, increase our understanding of the borderline between P and NP.

In the "opposite direction", there have been attempts to isolate subclasses of NP which exhibit dichotomies between P and NP-complete, i.e. non-trivial subclasses that do not admit NP-intermediate problems. For instance, Feder and Vardi [18] conjectured that the *constraint satisfaction problem* (CSP) over finite domains exhibits such a dichotomy, but so far the conjecture is only known to hold for domains of two and three elements, as proven by Schaefer [38] and Bulatov [7], respectively. One of the reasons behind this conjecture is that the constraint satisfaction problem is included in *monotone monadic SNP without inequality* (MMSNP), where SNP is a subset of NP characterizable through a special class of existential second-order sentences, and it is known that adding only marginally more expressive sentences to MMSNP results in a non-dichotomizable complexity class [18]. The set MMSNP is therefore viewed as a candidate as a maximal subclass of NP which does not contain any problems of intermediate complexity. From this it clearly follows that some restrictions, e.g. constraint language restrictions, are regarded as more interesting than removing arbitrary strings from the set of valid instances as in Ladner's original proof. In other words the existence of NP-intermediate subproblems depends heavily on which *parameter* one chooses to restrict and it is safe to say that we currently lack a deeper understanding of which parameters one can use to find NP-intermediate subproblems, and when dichotomies arise. Hence, we propose a strategy consisting of investigating subclasses of NP induced by different parameters, in order to determine which problems admit dichotomies and which do not, and increase our understanding of the puzzling nature of intermediate problems.

Article Structure

We begin (in Section 3) by presenting a diagonalization method for obtaining NP-intermediate problems, based on parameterizing decision problems in different ways. In our framework, a parameter, or a *measure function*, is simply a computable function ρ from the instances of some decision problem X to the non-empty subsets of $\mathbb{N}$. We say that such a function is *single-valued* if $\rho(I)$ is a singleton set for every instance of X, and *multi-valued* otherwise. Depending on the parameter one obtains problems with different characteristics. Simple applications of our method include the connection between the complexity class XP and NP-intermediate problems observed by Chen et al. [12]. Even though our method is still based on diagonalization we claim that the intermediate problems obtained are qualitatively different from the ones obtained by Ladner's original method, and that they can be used for gaining new insights into the complexity of computational problems. Whether a problem is "natural" or not is of course highly subjective and a matter of taste, but there is a wider consensus that some types of restrictions, such as constraint language restrictions, are more interesting than others. Throughout this article we will see that our diagonalization framework in combination with different measure functions allows us to construct NP-intermediate problems also for such non-trivial cases, which may constitute new and interesting sources of intermediate problems.

In Section 4, we analyze the applicability of the diagonalization method for single-valued measure functions. Under mild additional assumptions, we obtain a full understanding of when NP-intermediate problems arise when the measure function is single-valued and polynomial-time computable. We also relate the structure of subproblems induced by single-valued measure functions to the question of whether the set of all NP-intermediate problems is closed under disjoint union. Unfortunately, CSPs under structural restrictions (i.e. when considering instances with bounded width parameters) are not captured by these results since width parameters are typically not polynomial-time computable. To remedy this, we present a general method for obtaining NP-intermediate problems based on structurally restricted CSPs in Section 4.3. This is a generalization of a result by Grohe [22] who has shown that, under the assumption that FPT $\neq$ W[1], NP-intermediate CSP problems can be obtained by restricting the tree-width of their corresponding primal graphs. Our result implies that this holds also under the weaker assumption that P $\neq$ NP and for many width parameters. NP-intermediate problems based on structural restrictions have also been identified by Bodirsky & Grohe [5].

Multi-valued measure functions are apparently harder to study and a full understanding appears difficult to obtain. We first relate single-valued measure functions with multi-valued measure functions (in Section 5.1) and show that every multi-valued measure function effectively determines a single-valued measure function which shares many fundamental properties. Despite this, single-valued measure functions have limited applicability when studying problems parameterized by constraint languages, such as constraint satisfaction problems, and we give several examples which highlight why this is the case. For problems parameterized by constraint languages we therefore focus exclusively on multi-valued measure functions. Our first result (in Section 5.2) is inspired by Bodirsky & Grohe [5] who proved that there exists an infinite constraint language Γ over an infinite domain such that CSP(Γ) is NP-intermediate. We extend this and prove that whenever an infinite language Γ does not satisfy the so called *local-global* property, i.e. when CSP(Γ) $\notin$ P but CSP(Γ') $\in$ P for all finite $\Gamma' \subset \Gamma$, then there exists a language closely related to Γ such that the resulting CSP problem is NP-intermediate. The only requirement is that Γ can be extended by *extension operators* satisfying certain closure properties. Such an operator takes a set of relations as input and returns a superset (possibly infinite) with the property that any finite number of relations can be removed without affecting the expressive power of the language. We denote these as $\langle \cdot \rangle$ and provide two very different examples. The first operator $\langle \cdot \rangle_{pow}$ works for languages over both finite and infinite domains but gives relations of arbitrarily high arity. The second operator $\langle \cdot \rangle_{+}$ is limited to idempotent languages over infinite domains but does have the advantage that the arity of any relation is only increased by a small constant factor. Together with the language Γ° from Jonsson & Lööw [28] which does not satisfy the local-global property we are thus able to identify a concrete language $\langle \Gamma^{\circ} \rangle_{+}$ such that CSP($\langle \Gamma^{\circ} \rangle_{+}$) is NP-complete, CSP($\Gamma'$) $\in$ P for any finite $\Gamma' \subset \langle \Gamma^{\circ} \rangle_{+}$, and there exists a $\Gamma'' \subset \langle \Gamma^{\circ} \rangle_{+}$ such that CSP(Γ'') is NP-intermediate. The so-called *algebraic approach* [4, 8] has been very successful in studying the computational complexity of both finite- and infinite-domain CSPs. However, this approach is, to a large extent, limited to constraint languages that are finite. If one only considers tractable finite subsets of $\langle \Gamma^{\circ} \rangle_{+}$, we miss that there are both NP-intermediate and NP-complete problems within CSP($\langle \Gamma^{\circ} \rangle_{+}$). Hence, the constraint language $\langle \Gamma^{\circ} \rangle_{+}$ clearly shows that the algebraic approach in its present shape is not able to give a full understanding of CSP($\langle \Gamma^{\circ} \rangle_{+}$) and its subclasses.

Our second result (in Section 5.4) concerns the propositional *abduction* problem ABD(Γ). This problem can be viewed as a non-monotonic extension of propositional logic and it has numerous important applications ranging from automated diagnosis and text interpretation to planning. The complexity of propositional abduction has been intensively studied from a complexity-theoretic point of view (cf. [16, 35]) and the computational complexity is known for every finite Boolean constraint language Γ and many infinite languages [35]. In Nordh & Zanuttini [35], the question of whether such a classification is possible or impossible to obtain also for infinite languages was left open. Since the abduction problem can loosely be described as a combination of the SAT and

UNSAT problems, it might be expected that it, like the parameterized SAT($\cdot$) problem, does not contain any NP-intermediate problems. By exploiting our diagonalization method, we present a constraint language Γ such that $\textsc{Abd}(\Gamma)$ is NP-intermediate.

2. Preliminaries

For an arbitrary decision problem X, we let $I(X)$ denote its set of instances and $||I||$ denote the number of bits needed for representing $I \in I(X)$. By a *polynomial-time reduction* from problem X to problem X', we mean a Turing reduction from X to X' that runs in time $O(p(||I||))$ for some polynomial p. In other words the reduction may query an *oracle* for X' in order to solve X. We assume that the Turing machine performing the reduction has access to a specific oracle tape where it inputs the instance to be queried. Whenever convenient we actually utilize many-one reductions instead of Turing reductions since these are in some cases more natural.

A function f is said to be *computable* if it can be computed by a (universal) Turing machine. We remind the reader that the definitions of such functions can be recursive since a Turing machine has access to its own description due to Kleene's fixpoint theorem. Next, we define the concept of a *measure function* which is the cornerstone of the forthcoming diagonalization method.

Definition 1. Let X be a decision problem. A total and computable function $\rho : I(X) \to 2^{\mathbb{N}} \setminus \{\emptyset\}$ is said to be a *measure function.*

If $\rho(I)$ is a singleton set for every $I \in I(X)$, then we say that ρ is *single-valued*, and otherwise that it is *multi-valued*. We abuse notation in the first case and simply assume that $\rho : I(X) \to \mathbb{N}$. In addition a measure function ρ is said to be *polynomially bounded* if there exists a polynomial p such that $\rho(I) \leq p(||I||)$ for every $I \in I(X)$. This property is useful since we can write down $\rho(I)$ in unary in $p(||I||)$ time. The measure function ρ combined with a decision problem X yields a problem $X_\rho(S)$ parameterized by $S \subseteq \mathbb{N}$.

Instance: Instance I of X such that $\rho(I) \subseteq S$.
Question: Is I a yes-instance?

Note that $X_\rho(\mathbb{N})$ is equal to X for any measure function ρ.

Example 2. As an example we consider the Boolean satisfiability problem (SAT) and define two measure functions. Let I be an instance of SAT and define ρ_1 and ρ_2 such that $\rho_1(I)$ denotes the number of variables in I and $\rho_2(I) = \{ar(C) \mid C \text{ is a clause of } I\}$. Note that ρ_1 is single-valued while ρ_2 is multi-valued and that both are polynomial-time computable. Let $S = \{2k \mid k \in \mathbb{N}\}$. Clearly, $\text{SAT}_{\rho_1}(S)$ is the SAT problem restricted to an even number of variables, and $\text{SAT}_{\rho_2}(S)$ is the SAT problem restricted to instances with even clause length. The problems $\mathrm{SAT}_{\rho_1}(S)$ and $\mathrm{SAT}_{\rho_2}(S)$ are both NP-complete. However, we note that $\mathrm{SAT}_{\rho_1}(T)$ is in P whenever $T \subset S$ is finite while, for instance, $\mathrm{SAT}_{\rho_2}(\{k\})$ is NP-complete for all $k \geq 3$.

For more examples of both single- and multi-valued measure functions we refer the reader to Section 3.2. We now define one of the reoccurring decision problems in this article, the *constraint satisfaction problem*, which can be viewed as a generalization of SAT. Let Γ denote a (possibly infinite) set of finitary relations over some (possibly infinite) set D. We call Γ a *constraint language*. Given a relation $R \subseteq D^k$, we let $ar(R) = k$. The reader should note that we will sometimes express Boolean relations as conjunctions of Boolean clauses. The constraint satisfaction problem over Γ (abbreviated as $\mathrm{CSP}(\Gamma)$) is defined as follows.

Instance: A set V of variables and a set C of constraint applications $R(v_1, \ldots, v_k)$ where $R \in \Gamma$, $k = ar(R)$, and $v_1, \ldots, v_k \in V$.
Question: Is there a total function $f : V \rightarrow D$ such that $(f(v_1), \ldots, f(v_k)) \in R$ for each constraint $R(v_1, \ldots, v_k)$ in C?

For example let R_{NAE} be the following ternary relation on $\{0,1\}$: $R_{\mathrm{NAE}} = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. Then the well known NP-complete problem Not-All-Equal 3-Sat can be expressed as $\mathrm{CSP}(\{R_{\mathrm{NAE}}\})$. Similarly, if we define the relation $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$ then $\mathrm{CSP}(\{R_{1/3}\})$ corresponds to 1-in-3-SAT.

Finally, we prove a simple lemma regarding single-valued measure functions that will be important later on.

Lemma 3. *Let ρ be a single-valued and polynomial-time computable measure function. Let $S \subseteq \mathbb{N}$ and let T be a non-empty subset of S such that $S \setminus T = \{s_1, \ldots, s_k\}$. If $X_\rho(\{s_i\})$, $1 \leq i \leq k$, is in P, then there is a polynomial-time reduction from $X_\rho(S)$ to $X_\rho(T)$.*

Proof. Let I be an arbitrary instance of $X_\rho(S)$. Compute (in polynomial time) $\rho(I)$. If $\rho(I) \in \{s_1, \ldots, s_k\}$, then we can compute the correct answer in polynomial time. Otherwise, I is an instance of $X_\rho(T)$ and the reduction is trivial. □

3. Generation of NP-intermediate Problems

We will now extend Ladner's method to the parameterized problems in our framework. Section 3.1 contains the main result and Section 3.2 exemplifies both multi-valued and single-valued measure functions.

Theorem 4. *Let $X_\rho(\cdot)$ be a computational decision problem with a measure function ρ. Assume that $X_\rho(\cdot)$ and $S \subseteq \mathbb{N}$ satisfies the following properties:*

P0: $I(X)$ is recursively enumerable.
P1: $X_\rho(S)$ is NP-complete.
P2: $X_\rho(T)$ is in P whenever T is a finite subset of S.
P3: $X_\rho(S)$ is polynomial-time reducible to $X_\rho(T)$ whenever $T \subseteq S$ and $S \setminus T$ is finite.

If $\mathrm{P} \neq \mathrm{NP}$ *then there exists a set $S' \subset S$ such that $X_\rho(S')$ is in* $\mathrm{NP} \setminus \mathrm{P}$ *and $X_\rho(S)$ is not polynomial-time reducible to $X_\rho(S')$.*

The proof is an adaption of Papadimitriou's [37] proof where we use the abstract properties P0 – P3 instead of focusing on the size of instances. Papadimitriou's proof is, in turn, based on Ladner's original proof [30]. It may also be illuminating to compare with Schöning [39] and Bodirsky & Grohe [5]. Before the proof, we make some observations that will be used without explicit references. If ρ is single-valued and polynomial-time computable, then P2 implies P3 by Lemma 3. In many examples, $S = \mathbb{N}$ which means that P1 can be restated as NP-completeness of X. If P1 holds, then property P3 simply states that $X_\rho(T)$ is NP-complete for every cofinite $T \subseteq S$. Finally, we remind the reader that the polynomial-time bounds may depend on the choice of S in the definitions of P2 and P3. In the sequel, we let $X_\rho(\cdot)$ be a computational decision problem that together with $S \subseteq \mathbb{N}$ satisfies properties P0 – P3. Let A_X be an algorithm for $X_\rho(S)$, let $M_1, M_2, \ldots$ be an enumeration of all polynomial-time bounded deterministic Turing machines, and let $R_1, R_2, \ldots$ be an enumeration of all polynomial-time Turing reductions. Such enumerations are known to exist, cf. Papadimitriou [37].

The gist of the proof is to construct a function f which is used to create a problem which is too sparse to be NP-complete but too dense to be polynomial-time solvable. We define the function f by explicitly giving an algorithm that can be computed by a Turing machine F. This algorithm involves two distinct phases. In the first, for input n we compute a value k_n which is obtained by recursively computing f for $i = 1, 2, \ldots$. The final output, computed by a second phase of the algorithm, will either be k_n or $k_n + 1$. In this phase we choose one of two cases depending on whether k_n is even or odd. These rather complicated computations determine whether $X_\rho(S)$ is solvable in polynomial time for a large class of instances, or show that a polynomial time reduction is available for a large class of instances.

We finally use the fact that the problem of interest is NP-hard whilst all finite parametrizations are solvable in polynomial time, to show that the function f is strictly increasing. This will be enough to easily show that there exists a set $S_e \subset S$, defined using the function f, which results in a problem of NP-intermediate complexity. In order to bound the time taken by the calculation of f we make the Turing machine computing $f(n)$ to stop, in either phase, when it has taken more than n steps. This is easy to implement by introducing

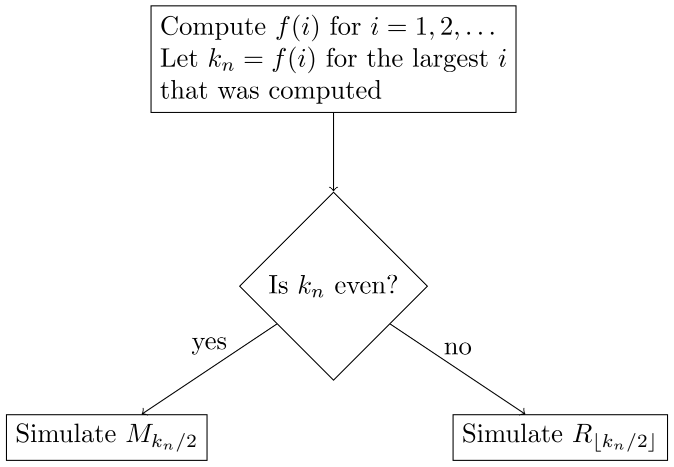
Compute $f(i)$ for $i = 1, 2, \ldots$ Let $k_n = f(i)$ for the largest i that was computed
Is k_n even?
yes
no
Simulate $M_{k_n/2}$
Simulate $R_{\lfloor k_n/2 \rfloor}$

Figure 1: A visualization of the two phases when computing $f(n)$. If the computation in any phase exceeds n steps then the machine stops and returns k_n.

a counter which is increased after every step in the computation. Before the formal definition of f we advise the reader to consult Figure 3.1 which is a simple diagram visualizing the flow of the Turing machine computing $f(n)$.

3.1.1. The first phase

First let $f(0) = f(1) = 0$. The computation of $f(n)$ starts with the computation of $f(0), f(1), f(2), \ldots,$ until the total number of steps F has used in computing this sequence exceeds n. Let i be the largest value for which F was able to completely compute $f(i)$ (during these n steps) and let $k_n = f(i)$.

3.1.2. The second phase

In the second phase of the execution of the machine F we have two cases depending on whether k_n is even or odd. In both cases, if this phase requires F to run for more than n computation steps, F stops and returns k_n (i.e., $f(n) = k_n$).

The even case

The first case is when k_n is even: here, F enumerates all instances I of $X_\rho(S)$ — this is possible by property P0. For each instance I, F simulates $M_{k_n/2}$ on the encoding of I, determines whether $A_X(I)$ is accepted, and finally, F computes f for all $x \in \rho(I)$. If $M_{k_n/2}$ rejects and $A_X(I)$ was accepted, and $f(x)$ is even for all $x \in \rho(I)$, then F returns $k_n + 1$ (i.e., $f(n) = k_n + 1$). Similarly, F also returns $k_n + 1$ if $M_{k_n/2}$ accepts and I is not accepted by A_X and $f(x)$ is even for all $x \in \rho(I)$.

The odd case

The second case is when k_n is odd. Again, F enumerates all instances I of $X_\rho(S)$. Let $E = \emptyset$. Now, for each instance I, F begins simulating $R_{\lfloor k_n/2 \rfloor}$ on the encoding of I with an oracle for A_X. Whenever the simulation notices that $R_{\lfloor k_n/2 \rfloor}$ enters an oracle state, we calculate $\rho(I') = E'$ (where I' is the $X_\rho(S)$ instance corresponding to the input of the oracle tape), and add the members of E' to E. When the simulation is finished we first calculate $f(x)$ for every $x \in E$. If the result of any $f(x)$ operation is odd we return $k_n + 1$. We then compare the result of the reduction with $A_X(I)$. If the results do not match, i.e. if one is accepted or rejected while the other is not, we return $k_n + 1$.

3.1.3. Wrapping up

This completes the definition of f. Note that f can be computed in polynomial time (regardless of the time complexity of computing ρ and A_X) since the input is given in unary. We now use the function f to define our intermediate language. Let

$$S_e = \{x \mid x \in S \text{ and } f(x) \text{ is even}\}.$$

In other words the function f is used to blow holes in S, and the holes, i.e. the removed elements, are determined on the basis of whether $f(x)$ is even or odd.

Lemma 5. *The function f is increasing and unbounded: for all $n \geq 0$, $f(n) \leq f(n+1)$ and $\{f(n) \mid n \in \mathbb{N}\}$ is infinite, unless* P = NP.

Proof. We first prove by induction that $f(n) \leq f(n+1)$ for all $n \geq 0$. This obviously holds for $n = 0$ and $n = 1$. Assume that this holds for an arbitrary number $i > 1$. In the first phase of the computation of $f(i)$ the Turing machine F computes $f(i')$ for all $i' < i$. Let k_i and k_{i+1} be the largest values that was successfully computed within i and $i+1$ steps, respectively. Clearly $k_{i+1} \geq k_i$ since the only difference is that we in the latter case can perform one more computation. In the second phase of the computation of $f(i)$ the Turing machine F returns either k_i or $k_i + 1$. There are two cases to consider. If $k_i = k_{i+1}$ then F will simulate the same Turing machine M_{k_i} or the same reduction $R_{\lfloor k_i/2 \rfloor}$ in the computation of $f(i+1)$ — hence $f(i+1) \geq f(i)$. In the second case where $k_{i+1} > k_i$ the result follows directly since $k_{i+1} \geq k_i + 1$.

We continue by showing that there is no n_0 such that $f(n) = k_{n_0}$ for all $n > n_0$ unless P = NP. If there is such an n_0, then there is also an n_1 such that for all $n > n_1$, the value k_n computed in the first phase is k_{n_0}. If k_{n_0} is even, then on all inputs $n > n_1$ the machine $M_{k_{n_0}/2}$ correctly decides $X_\rho(S_e)$ and thus $X_\rho(S_e)$ is in P. But since $f(n) = k_{n_0}$ for all $n > n_1$, we have that $S \setminus S_e$ is finite, and thus $X_\rho(S)$ is polynomial-time reducible to $X_\rho(S_e)$ by Property P3, which is a contradiction since $X_\rho(S)$ is NP-complete by Property P1. Similarly, if k_{n_0} is odd, then on all inputs $n > n_1$ the function $R_{\lfloor k_{n_0}/2 \rfloor}$ is a valid reduction from $X_\rho(S)$ to $X_\rho(S_e)$ and thus $X_\rho(S_e) \notin$ P. But since $f(n) = k_{n_0}$ for all $n > n_1$, we have that S_e is finite, and we conclude that $X_\rho(S_e)$ is in P by Property P2, which is a contradiction since $X_\rho(S)$ is NP-complete by Property P1. □

Proof of Theorem 4

We conclude the proof by showing that $X_\rho(S_e)$ is neither in P, nor is $X_\rho(S)$ polynomial-time reducible to $X_\rho(S_e)$, unless P = NP. By Property P1, $X_\rho(S_e)$

is in NP since $S_e \subseteq S$. Assume now that $X_\rho(S_e)$ is in P. Then there is an i such that M_i solves $X_\rho(S_e)$. Thus, by the definition of f, there is an n_1 such that for all $n > n_1$ we have $f(n) = 2i$; this contradicts that f is increasing. Similarly, assume that $X_\rho(S)$ is polynomial-time reducible to $X_\rho(S_e)$. Then, there is an i such that R_i is a polynomial-time reduction from $X_\rho(S)$ to $X_\rho(S_e)$. It follows from the definition of f that there is an n_1 such that $f(n) = 2i-1$ for all $n > n_1$, and this contradicts that f is increasing. $\square$

It also follows from the proof that property P1 (i.e. the NP-hardness of the original problem) can be replaced by hardness for other complexity classes within NP. By noting that $X_\rho(S_e)$ is recursively enumerable, we obtain the following corollary.

Corollary 6. *Let $X_\rho(\cdot)$ be a computational decision problem with a measure function ρ such that $X_\rho(\cdot)$ and $S \subseteq \mathbb{N}$ satisfies properties P0–P3 in Theorem 4. Let $S = T_1$. Then there exists an infinite chain $T_1 \supset T_2 \supset \dots$ such that $X_\rho(T_i)$ is not in P and $X_\rho(T_i)$ is not polynomial-time reducible to $X_\rho(T_{i+1})$ for any $i \geq 1$.*

3.2. Examples

We close this section with two examples illustrating single-valued and multi-valued measure functions. We first see that Ladner's result is a straightforward consequence of Theorem 4.

Example 7. Let X be an NP-complete problem such that $I(X)$ is recursively enumerable. For an arbitrary instance $I \in I(X)$, we let the single-valued measure function ρ be defined such that $\rho(I) = ||I||$. We verify that $X_\rho(\mathbb{N})$ satisfies properties P0 – P3 and conclude that there exists a set $T \subseteq \mathbb{N}$ such that $X_\rho(T)$ is NP-intermediate. Properties P0 and P1 hold by assumption and property P2 holds since $X_\rho(U)$ can be solved in constant time whenever U is finite. If $U \subseteq \mathbb{N}$ and $\mathbb{N} \setminus U = \{x_1, \dots, x_k\}$, then $X_\rho(\{x_i\})$, $1 \leq i \leq k$, is solvable in constant time and we can apply Lemma 3. Thus, property P3 holds, too.

If we briefly return to Example 7 from Section 2 where X = SAT and where $\rho_1(I)$ returns the number of variables in I, then by recapitulating the reasoning in the above example it is easy to show that there exists a set $S \subset \mathbb{N}$ such that $\text{SAT}_{\rho_1}(S)$ is NP-intermediate. Note however that same reasoning cannot be applied when ρ_2 is the multi-valued measure function returning the set consisting of the lengths of all clauses in an instance, since $\text{SAT}_{\rho_2}(\mathbb{N})$ does not satisfy property P2.

Example 8. To illustrate multi-valued measure functions, we turn our attention to the SUBSET-SUM problem [29].

INSTANCE: A finite set $Y \subseteq \mathbb{N}$ and a number $k \in \mathbb{N}$.
QUESTION: Is there a $Y' \subseteq Y$ such that $\sum Y' = k$?

We define a multi-valued measure function by letting $\rho((Y, k)) = Y$. Once again, properties P0 and P1 hold by assumption so it is sufficient to prove that SUBSET-SUM$_\rho(\mathbb{N})$ satisfies P2 and P3. Property P2: instances of SUBSET-SUM can be solved in time $O(\text{poly}(||I||) \cdot c(I))$, where $c(I)$ denotes the difference between the largest and smallest number in Y [19]. This difference is finite whenever we consider instances of SUBSET-SUM$_\rho(S)$ where $S \subseteq \mathbb{N}$ is finite. Property P3: arbitrarily choose $S \subseteq \mathbb{N}$ such that $\mathbb{N} \setminus S$ is finite. We present a polynomial-time Turing reduction from SUBSET-SUM$_\rho(\mathbb{N})$ to SUBSET-SUM$_\rho(S)$. Let $I = (Y, k)$ be an instance of SUBSET-SUM$_\rho(\mathbb{N})$. Let $T = Y \setminus S$, i.e. the elements of the instance which are not members of the smaller set S. Since $\mathbb{N} \setminus S$ is finite, T is a finite set, too. Let $Z = Y \cap S$. For every subset $T'_i = \{x_1, \ldots, x_{im}\}$ of T, we let $I'_i = (Z, k'_i)$, where $k'_i = k - (x_1 + \ldots + x_{im})$. Then, it is easy to see that I is a yes-instance if and only if at least one I'_i is a yes-instance. Finally, we note that the reduction runs in time $O(\text{poly}(||I||) \cdot 2^c)$, where $c = |\mathbb{N} \setminus S|$, and this is consequently a polynomial-time reduction for every fixed S.

We see that the existence of a pseudo-polynomial algorithm and the possibility to perform auto-reductions are crucial in the above example but that not much more is needed. Hence, there are many other pseudo-polynomial problems that can be used instead of SUBSET-SUM. Several examples that are closely related to the example above can be found in Papadimitriou [36]. Another interesting source of pseudo-polynomial problems are those with polynomial-time approximation schemes: it is known that every "well-behaved" problem that has a fully polynomial-time approximation scheme can be solved in pseudo-polynomial time. For details, see Garey & Johnson [19].

4. Single-Valued Measure Functions

Single-valued measure functions have been studied in the literature before. For instance, Chen et al. [12] have discovered a striking connection between NP-intermediate problems and the parameterized complexity class XP (XP denotes the class of decision problems X that are solvable in time $O(||I||^{f(k)})$ for some polynomial-time computable parameter k and some computable function f). Such a connection can be established via Theorem 4, too. Chen et al. demand that the measure function ρ can be computed in polynomial time, which gives the following result.

Proposition 9. *Let X be a decision problem and ρ a polynomial-time computable single-valued measure function such that $X_\rho(\cdot)$ satisfies properties P0 and P1, and $X_\rho \in$ XP. Then there exists a $T \subseteq \mathbb{N}$ such that $X_\rho(T)$ is NP-intermediate.*

PROOF. We note that $X_\rho(S)$ is in P whenever S is a finite subset of $\mathbb{N}$. Hence, X_ρ satisfies P2 and consequently P3. The result follows from Theorem 4. □

The above proposition can also be used to construct infinite descending chains of NP-intermediate problems which strongly mirrors the results from

Chen et al. The remainder of the section is divided into three parts: Section 4.1 is concerned with properties of polynomial-time computable single-valued measure functions, Section 4.2 studies the complexity of subproblems induced by polynomial-time computable measure single-valued measure functions in greater detail, and Section 4.3 is concerned with structurally restricted CSPs.

4.1. Polynomial-Time Computable Measure Functions

By Theorem 4, we know that properties P0 – P3 are sufficient to assure the existence of NP-intermediate problems. A related question is to what degree the properties are also necessary. Here, we investigate the scenario when P2 and P3 do not necessarily hold.

Theorem 10. *Assume X is a decision problem and ρ a polynomial-time computable single-valued measure function such that $X_\rho(\mathbb{N})$ satisfies P0 and P1. Let $S_P = \{s \in \mathbb{N} \mid X_\rho(\{s\}) \in \mathrm{P}\}$ and assume membership in S_P is a decidable problem. Then, at least one of the following holds:*

1. *there exists a set $T \subseteq S_P$ such that $X_\rho(T)$ is NP-intermediate,*
2. *there exists a $t \in \mathbb{N}$ such that $X_\rho(\{t\})$ is NP-intermediate, or*
3. *X_ρ admits no NP-intermediate subproblems.*

Proof. If $X_\rho(\{s\})$ is NP-complete for every $s \in \mathbb{N}$, then we are in case (3) so we assume this is not the case. If there exists $s \in \mathbb{N}$ such that $X_\rho(\{s\})$ is NP-intermediate, then we are in case (2) so we assume this does not hold either. Thus, we may henceforth assume that there exists $s \in \mathbb{N}$ such that $X_\rho(\{s\}) \in P$ and that $X_\rho(\{u\})$ is NP-complete whenever $u \in \mathbb{N} \setminus S_P$. This implies that S_P is non-empty. Once again, we single out two straightforward cases: if $X_\rho(S_P)$ is NP-intermediate, then we are in case (1), and if $X_\rho(S_P)$ is in P, then we are in case (3) (since $X_\rho(\{u\})$ is NP-complete whenever $u \notin S_P$). Hence, we may assume that $X_\rho(S_P)$ is NP-complete (note that $X_\rho(S_P) \in \mathrm{NP}$ since $X_\rho(\mathbb{N}) \in \mathrm{NP}$ by P1), i.e. $X_\rho(S_P)$ satisfies P1. Furthermore, $X_\rho(S_P)$ satisfies P0 since S_P is a decidable set and the instances of X are recursively enumerable. To generate the instances of $X_\rho(S_P)$, we generate the instances of X one after another and output instance I if and only if $\rho(I)$ is in S_P.

We finally show that $X_\rho(S_P)$ satisfies P2 and P3. By Lemma 3 it is sufficient to prove that $X_\rho(S_P)$ satisfies P2 since ρ is single-valued and polynomial-time computable. Assume there exists a finite set $K \subseteq S_P$ such that $X_\rho(K) \notin \mathrm{P}$. Let $\emptyset \subset K' \subseteq K$ be a subset such that $X_\rho(K')$ is a member of P; such a set exists since $K \subseteq S_P$. For every $k' \in K'$, we know that $X_\rho(\{k'\}) \in \mathrm{P}$. Hence, we can apply Lemma 3 and deduce that there exists a polynomial-time reduction from $X_\rho(K)$ to $X_\rho(K')$. This contradicts the fact that $X_\rho(K)$ is not a polynomial-time solvable problem. We can now apply Theorem 4 and conclude that there exists a set $T \subseteq S_P$ such that $X_\rho(T)$ is NP-intermediate, i.e. we are in case (1). □

Problems parameterized by multi-valued measure functions are apparently very different from those parameterized by single-valued functions. For instance, Lemma 3 breaks down which indicates that the proof strategy used in Theorem 10 is far from sufficient to attack the multi-valued case.

4.2. Complexity of Subproblems Induced by Complementary Sets

In this section we study the complexity of problems of the form $X_\rho(\mathbb{N} \setminus T)$, where $T \subset \mathbb{N}$ and ρ is a single-valued measure function such that $X_\rho(T)$ is NP-intermediate. This question is connected to the complexity of unions of disjoint sets in NP and we note that this is a problem that has attracted considerable attention in the literature, cf. [21, 40]. We basically show the following: if ρ and T satisfy certain conditions, then either $X_\rho(\mathbb{N} \setminus T)$ is NP-complete, or the set of all NP-intermediate problems is not closed under disjoint unions. We also show that if T is subject to some mild additional restrictions, then $X_\rho(\mathbb{N} \setminus T)$ is NP-complete unconditionally. To describe the results in more detail, let $\mathbf{T} = \{T_1, T_2, \ldots\}$ denote the subsets of $\mathbb{N}$ such that membership in T_i, $i \geq 1$, can be decided in polynomial time for integers written in unary.

Proposition 11. *Let X be an NP-complete problem and ρ a polynomial-time computable and polynomially bounded single-valued measure function on X. Then one of the following hold:*

1. *for every $T \in \mathbf{T}$, if $X_\rho(T)$ is NP-intermediate, then $X_\rho(\mathbb{N} \setminus T)$ is NP-complete, or*
2. *the set of all NP-intermediate problems is not closed under disjoint union.*

Proof. Assume to the contrary that the set of all NP-intermediate problems is closed under disjoint union and that there exists a set $T \in \mathbf{T}$ such that $X_\rho(\mathbb{N} \setminus T)$ is not NP-complete. We first show that the problem $X_\rho(\mathbb{N} \setminus T)$ cannot be polynomial-time solvable. Assume to the contrary that there exists a polynomial-time algorithm A for $X_\rho(\mathbb{N} \setminus T)$. We show that there exists a polynomial-time many-one reduction from the NP-complete problem X to $X_\rho(T)$. This leads to a contradiction since $X_\rho(T)$ in NP-intermediate.

The reduction goes as follows: let $I_{\text{yes}}, I_{\text{no}}$ be arbitrary yes- and no-instances of $X_\rho(T)$, respectively. These are required since we are performing a many-ony reduction. Given an instance I of X, do the following.

1. let $y = \rho(I)$,
2. let x be y written in unary,
3. if $x \in \mathbb{N} \setminus T$, then use algorithm A for checking whether I is a yes-instance of X or not. If this is the case, then output I_{yes} and, otherwise, output I_{no} and stop,
4. output I.

It is easy to verify that this procedure is a reduction from X to $X_\rho(T)$ and, furthermore, that it runs in polynomial time. By assumption, y can be computed in polynomial time and we can compute x in polynomial time, too, since ρ is polynomially bounded. The test in line 3 can be performed in polynomial time due to the choice of T and we have, additionally, assumed that algorithm A runs in polynomial time.

We have now verified that both $X_\rho(T)$ and $X_\rho(\mathbb{N} \setminus T)$ are NP-intermediate problems. These two problems are disjoint and their union (which equals $X_\rho(\mathbb{N})$ since ρ is single-valued) is clearly NP-complete since $X = X_\rho(\mathbb{N})$. $\Box$

By making some additional assumptions we can make the above proposition more precise.

Proposition 12. *Let X be an NP-complete problem and ρ a polynomial-time computable and polynomially bounded single-valued measure function on X. Assume the following hold:*

1. *$X_\rho(T)$ is NP-intermediate for some $T \in \mathbf{T}$ and*
2. *for each instance $I \in I(X)$, one can in polynomial time compute an instance I^+ such that $\rho(I^+) = \rho(I) + 1$ and I^+ is a yes-instance if and only if I is a yes-instance.*

If $x \in T$ implies $x + 1 \notin T$, then $X_\rho(\mathbb{N} \setminus T)$ is NP-complete. Otherwise, there exists a set $T' \subseteq T$ such that $X_\rho(T')$ is NP-intermediate and $X_\rho(\mathbb{N} \setminus T')$ is NP-complete. The set T' can always be assumed to equal either $T_o = \{t \mid t \in T \text{ is odd}\}$ or $T_e = \{t \mid t \in T \text{ is even}\}$.

Proof. We first assume that if $x \in T$, then $x+1 \notin T$. We present a polynomial-time reduction from X to $X_\rho(\mathbb{N} \setminus T)$. Let I be an arbitrary instance of X. Compute (in polynomial-time) $\rho(I)$. Since ρ is polynomially bounded we can in polynomial time write down $\rho(I)$ in unary. Let x denote this string. Next, we check (in polynomial time) whether $x \in T$ or not. If $x \notin T$, then the result of the reduction is I itself. Otherwise, compute (in polynomial time) I^+ and output this instance. We see that this reduction is indeed polynomial-time computable. Furthermore, it is a reduction from X to $X_\rho(\mathbb{N} \setminus T)$ — it is sufficient to note that $\rho(I^+) = \rho(I) + 1$ and $\rho(I) + 1 \notin T$. Finally, we know that I^+ is a yes-instance if and only if I is a yes-instance.

Assume now that there exists $x \in \mathbb{N}$ such that $\{x, x+1\} \subseteq T$. Obviously, $X_\rho(T_o)$ and $X_\rho(T_e)$ are not NP-complete problems since $X_\rho(T)$ is not NP-complete. Assume both problems are in P. Then, we claim $X_\rho(T)$ is in P, too, which leads to a contradiction. Given an instance I of $X_\rho(T)$, compute (in polynomial time) $\rho(I)$. Next, check whether $\rho(I)$ is odd or not. If so, then apply the polynomial-time algorithm for $X_\rho(T_o)$ and, otherwise, the polynomial-time algorithm for $X_\rho(T_e)$. We conclude that at least one of $X_\rho(T_o)$ and $X_\rho(T_e)$ is NP-intermediate, and we choose T' such that T' equals either T_o or T_e and $X_\rho(T')$ is NP-intermediate. Note that if $x \in T'$, then $x + 1 \notin T'$. Also note that given $x \in \mathbb{N}$ in unary, it is polynomial-time decidable whether $x \in T'$ (since we can check whether $x \in T$ or not). The first part of the proof immediately gives the result. $\Box$

4.3. Structurally Restricted CSP*s*

When identifying tractable (i.e. polynomial-time solvable) fragments of constraint satisfaction problems and similar problems, two main types of results have been considered in the literature. The first one is to identify constraint languages Γ such that $\mathrm{CSP}(\Gamma) \in \mathrm{P}$, and the second one is to restrict the structure induced by the constraints on the variables. The second case is often concerned with associating some structure with each instance and then identifying sets of structures that yield tractable problems. The classical example of this approach is to study the *primal graph* or *hypergraph* of CSP instances. Given a CSP instance I with variable set V, we define its primal graph $G = (V, E)$ such that $(v_i, v_j) \in E$ if and only if variables v_i, v_j occur simultaneously in some constraint, and we define the hypergraph $\mathcal{H} = (V, \mathcal{E})$ such that the hyperedge $\{v_{i_1}, \ldots, v_{i_k}\} \in \mathcal{E}$ if and only if there is a constraint $R(v_{i_1}, \ldots, v_{i_k})$ in I.

When it comes to defining structurally restricted problems that are tractable, one is typically interested in certain parameters of these (hyper)graphs such as *tree-width*, *fractional hypertree width* [23], or *submodular width* [34]. It is, for instance, known that any finite-domain CSP instance I with primal graph $G = (V, E)$ can be solved in $||I||^{O(\mathrm{tw}(G))}$ time [14] where $\mathrm{tw}(G)$ denotes the tree-width of G, and it can be solved in $||I||^{O(\mathrm{fhw}(\mathcal{H}))}$ time [23] where $\mathrm{fhw}(\mathcal{H})$ denotes the fractional hypertree width of $\mathcal{H}$. Since these results rely on the domains being finite, we restrict ourselves to finite-domain CSPs throughout this section. Now note that if given a finite constraint language Γ, then the instances of $\mathrm{CSP}(\Gamma)$ are recursively enumerable and $\mathrm{CSP}(\Gamma)$ is in NP. If Γ is infinite, then this is not so evident and it may, in fact, depend on the representation of relations. We adopt a simplistic approach and assume that it is decidable to check whether a relation is included in Γ, given that it is represented as a list of tuples. Under this assumption the instances of $\mathrm{CSP}(\Gamma)$ are recursively enumerable also for infinite Γ.

If we use tw and fhw as measure functions then the resulting problems $\mathrm{CSP}_{\mathrm{tw}}$ and $\mathrm{CSP}_{\mathrm{fhw}}$[1] satisfy property P2. To see this simply note that if the tree-width or fractional hypertree width is restricted by k then such CSP instances can be solved in $||I||^{O(k)}$ time [14, 23]. If the width parameter under consideration is polynomial-time computable, then we have property P3 (via Lemma 3), too, and conclude that NP-intermediate fragments exist. Unfortunately, this is typically not the case. It is for instance NP-complete to determine whether a given graph G has treewidth at most k or not [2] if k is part of the input. This is a common feature that holds for, or is suspected to hold for, many width parameters. Hence, width parameters are a natural source of single-valued measure functions that are not polynomial-time computable. Such measure functions are problematic since we cannot prove the existence of NP-intermediate subproblems by using simplifying results like Proposition 9 or Theorem 10. By a few additional assumptions we can however still prove the applicability of Theorem 4. Note that if k is fixed, and thus not part of the input, then the graphs

[1]We slightly abuse notation since tw and fhw are not directly defined on problem instances.

with tree-width $\leq k$ can be recognized in linear time [6]. This is not uncommon when studying width parameters — determining the width exactly is computationally hard but it can be computed or estimated in polynomial time under additional assumptions. We arrive at the following result.

Proposition 13. *Assume that X is a decision problem and ρ is a single-valued measure function such that $X_\rho(\cdot)$ satisfies properties P0 and P1. Furthermore, suppose that for each set $\{0, \ldots, k\}$ there exists a promise algorithm A_k for $X_\rho(\{0, \ldots, k\})$ with the following properties:*

- *if $\rho(I) \leq k$, then A_k returns the correct answer in $p_k(||I||)$ steps, where p_k is a polynomial only depending on k, and*
- *if $\rho(I) > k$, then A_k either returns a correct answer or does not answer at all.*

Then there exists a set $S \subset \mathbb{N}$ such that $X_\rho(S)$ is NP-intermediate.

Proof. Let X^k denote the computational problem X restricted to instances $I \in I(X)$ such that $\rho(I) \geq k$. Assume there exists a k such that $X^k \in P$ and let B be an algorithm for this problem running in time $q(||I||)$ for some polynomial q. For $X_\rho(\{0, \ldots, k-1\})$, we have algorithm A_{k-1} described above. Given an arbitrary instance I of X, we may not be able to compute $\rho(I)$ and choose which algorithm to run. Do as follows: run algorithm A_{k-1} for $p_{k-1}(||I||)$ steps on input I. If A_{k-1} produces an answer, then this is correct. If A_{k-1} does not produce an answer, then we know that $\rho(I) > k-1$ and we can apply algorithm B. All in all, this takes $O(p_{k-1}(||I||) + q(||I||))$ time so $X \in$ P which leads to a contradiction.

If X^k is NP-intermediate for some k, then we simply let $S = \{k, k+1, \ldots\}$. We can henceforth assume that X^k is NP-complete for all k. Obviously, $X_\rho(\mathbb{N})$ satisfies property P2 since algorithm A_k, $k \geq 0$, runs in polynomial time. We show that it satisfies property P3, too. Let $T \subseteq \mathbb{N}$ be a finite set and let $m = \max T$. We know that X^{m+1} is NP-complete. Hence, there exists a polynomial-time reduction from the NP-complete problem $X_\rho(\mathbb{N})$ to X^{m+1} which, in turn, admits a trivial polynomial-time reduction to $X_\rho(\mathbb{N} \setminus T)$ since $\{m+1, m+2, \ldots\} \subseteq \mathbb{N} \setminus T$. We can now apply Theorem 4 and obtain the set S. □

We apply this result to CSP_{tw} and CSP_{fhw}, respectively. Clearly, both these problems satisfy properties P0 and P1 due to the assumptions that we have made. For CSP_{tw}, we let A_k work as follows: given a CSP instance I, check whether I has treewidth $\leq k$ using Bodlaender's [6] algorithm. If the algorithm answers "no", then go into an infinite loop. Otherwise, decide whether I has a solution or not in $||I||^{O(k)}$ time. Proposition 13 implies that there exists a set $T \subseteq \mathbb{N}$ such that $\text{CSP}_{\text{tw}}(T)$ is NP-intermediate. We observe that Grohe [22] has shown a similar result under the assumption that FPT $\neq$ W[1] instead of P $\neq$ NP. Many other width parameters can also be used for obtaining NP-intermediate problems. One example is CSP_{fhw} for which the proof is very similar but is instead based on Theorem 4.1 in Marx [33].

Theorem 14. *Given a hypergraph $\mathcal{H}$ and a rational number $w \geq 1$, it is possible in time $||H||^{O(w^3)}$ to either*

- *compute a fractional hypertree decomposition of $\mathcal{H}$ with width at most $7w^3 + 31w + 7$, or*
- *correctly conclude that the fractional hypertree width of $\mathcal{H}$ is strictly greater than w.*

Let A_k work as follows. Given a CSP instance I, apply Marx's approximative algorithm with $w = k$. If the algorithm concludes that the fractional hypertree width is larger than k, then go into an infinite loop. Otherwise, compute the solution (by using the algorithm by Grohe and Marx [23]) of I in $||I||^{O(k^3)}$ time using the decomposition produced by the algorithm. We conclude, by Proposition 13, that there exists a set $T \subseteq \mathbb{N}$ such that $\mathrm{CSP}_{\mathrm{fhw}}(T)$ is NP-intermediate.

We finally note that one does not need to consider the full CSP problem (i.e. where all relations are allowed) when constructing NP-intermediate problems. To exemplify, let $\mathrm{CSP}(\mathcal{C}, \Gamma)$ denote the $\mathrm{CSP}(\Gamma)$ problem restricted to instances I such that the primal graph of I is a member of $\mathcal{C}$. Arbitrarily choose a constraint language Γ such that $\mathrm{CSP}(\Gamma)$ is NP-complete and arbitrarily choose a set $T \subseteq \mathbb{N}$ such that $\mathrm{CSP}_{\mathrm{tw}}(T)$ is NP-intermediate. It is now easy to use Proposition 13 and prove that the problem $\mathrm{CSP}(\{G \mid G \text{ is a graph and } \mathrm{tw}(G) \in T\}, \Gamma)$ is NP-intermediate. This basic idea can be varied in many different ways in order to obtain various NP-intermediate CSP problems.

5. Multi-Valued Measure Functions

In the following sections we turn our attention to multi-valued measure functions and apply them to constraint problems. The structure is as follows: in Section 5.1 we describe the relationship between multi-valued and single-valued measure functions and explain why multi-valued measure functions are preferable when working with constraint language restrictions, in Sections 5.2 and 5.3 we investigate constraint satisfaction problems and provide sufficient conditions for the existence of intermediate problems for both finite and infinite domains, and in Section 5.4 we use our framework to construct an NP-intermediate Boolean abduction problem.

5.1. Multi-Valued Measure Functions Compared to Single-Valued Measure Functions

To see why multi-valued measure functions are preferable when working with constraint language restrictions, consider the following illustrative case: given a constraint satisfaction problem parameterized with a constraint language Γ, let ρ denote the single-valued measure function defined to return the highest arity of any constraint in a given instance: $\rho((V, C)) = \max\{k \mid R(v_1, \ldots, v_k) \in C\}$. Let $\mathrm{CSP}^*_\rho(X)$ denote the $\mathrm{CSP}(\Gamma)$ problem restricted to instances I such that

$\rho(I) \in X$, and assume there exists a set $X \subset \mathbb{N}$ such that $\mathrm{CSP}^*_\rho(X)$ is NP-intermediate. Can we from this conclude that there exists a constraint language $\Gamma' \subset \Gamma$ such that $\mathrm{CSP}(\Gamma')$ is NP-intermediate? In general, the answer is no since the set of valid instances of $\mathrm{CSP}^*_\rho(X)$ are not in a one-to-one correspondence with any constraint language restriction. Note that $\mathrm{CSP}^*_\rho(X)$ is not the same problem as $\mathrm{CSP}(\{R \in \Gamma \mid ar(R) \in X\})$. If we on the other hand define the multi-valued measure function $\sigma((V, C)) = \{k \mid R(v_1, \ldots, v_k) \in C\}$, then for every $X \subset \mathbb{N}$ the problem $\mathrm{CSP}^*_\sigma(X)$ is equivalent to $\mathrm{CSP}(\{R \in \Gamma \mid \mathrm{ar}(R) \in X\})$.

Multi-valued measure functions are in general therefore more powerful than single-valued measure function since they are more closely related to e.g. constraint language restrictions. For every multi-valued measure function it is however possible to construct a single-valued measure function which preserves some of the properties of the original function. The intuition behind this is that every set $\rho(I)$ (which is finite since a measure function by definition is computable) with a suitable encoding can be associated with a natural number x. Then define the single-valued function ρ' as $\rho'(I) = x$, and let the set S' be defined so that every $x \in S'$ corresponds to a finite subset of S. As will be clear later on $X_{\rho'}(S')$ then defines the same set of instances and satisfies properties P0–P3 if $X_\rho(S)$ satisfies these, but it is a one-way street in the sense that blowing holes into S' with ρ' does not necessarily allow one to do the same thing with S and ρ. To make this more precise we formalize the notion of encoding a finite set as a number. Let $\mathcal{F}$ denote the set of all finite subsets of $\mathbb{N}$

Definition 15. A function $f : \mathcal{F} \mapsto \mathbb{N}$ is a *coding function* if (1) $f(x)$ is computable in polynomial time for any $x \in \mathcal{F}$ and (2) f is injective.

For any multi-valued measure function ρ and coding function f one can then define a single-valued measure function $\rho_f(I) = f(\rho(I))$ for any $I \in I(X)$.

Proposition 16. *Let $X_\rho(\cdot)$ be a computational decision problem with a multi-valued, polynomial-time computable measure function ρ such that $X_\rho(\cdot)$ and $S \subseteq \mathbb{N}$ satisfies properties P0 – P2. Then for any coding function f there exists an $S_f \subseteq \mathbb{N}$ such that (1) $I(X_{\rho_f}(S_f)) = I(X_\rho(S))$ and (2) $X_{\rho_f}(S_f)$ satisfies properties P0 – P3.*

Proof. Let $S_f = \{f(\{x_1, \ldots, x_k\}) \mid \{x_1, \ldots, x_k\} \subseteq S\}$. By construction it then follows that $\rho(I) \subseteq S$ if and only if $\rho_f(I) \in S_f$ for any $I \in I(X)$. Thus $X_{\rho_f}(S_f)$ satisfies P0 and P1. As for property P2 let T be a finite subset of S_f and define $T' = \bigcup_{x \in T} f^{-1}(x)$. We can then reduce any instance I of $X_{\rho_f}(T)$ to $X_\rho(T')$, which is solvable in polynomial time by assumption. Last, by Lemma 3, $X_{\rho_f}(S_f)$ satisfies property P3 since ρ_f is computable in polynomial time. $\square$

There is no shortage of coding functions, but some careful steps must be taken to ensure that it is polynomial-time computable. If we for instance use a standard textbook Gödel coding g and encode each finite set $\{x_1, \ldots, x_k\}$ as $g(\{x_1, \ldots, x_k\}) = p_1^{x_1} \ldots p_k^{x_k}$, where p_i denotes the ith prime number, then g

is injective but is not known to be polynomial-time computable (if input size is taken to be the number of bits required to represent the set of numbers). Instead one can for example define f to return the number corresponding to a binary encoding of a set $\{x_1, \ldots, x_k\}$ (demarked in a suitable way). For every decision problem X, every multi-valued measure function ρ and set of numbers S it is hence possible to find a closely related polynomial-time computable single-valued measure function ρ' and a set of numbers S', such that $I(X_\rho(S)) = I(X_{\rho'}(S'))$. Note however that if $T \subset S$ then there does not necessarily exist a $T' \subset S'$ such that $I(X_\rho(T)) = I(X_{\rho'}(T'))$. For constraint language restrictions such as the ones in Sections 5.2, 5.3 and 5.4 we therefore still need to use multi-valued measure functions.

For single-valued measure functions it is also possible to relate the complexity between a problem and its subproblems induced by the measure function. We give a straightforward proposition illustrating this and sketch why the same techniques appears infeasible to handle the multi-valued case. A *complexity parameter* for a decision problem X is a polynomia-time computable and polynomially bounded function m from $I(X)$ to $\mathbb{N}$. Natural choices for e.g. SAT is the number of variables or the number of clauses. In the sequel we assume that m is a complexity parameter for the decision problem X. Let

$$C_{X,m} = \inf\{c \mid \text{there is an algorithm } A \text{ for } X \text{ running in time } c^{m(I)}\}.$$

A problem X might not necessarily be solvable in time $c^{m(I)}$ but still solvable in time $(c+\epsilon)^{m(I)}$ for every $\epsilon > 0$. When X is solvable in time $c^{m(I)}$ for all $c > 0$, we say that X is a *subexponential problem.* For $X =$ 3-SAT the conjecture that $C_{X,m} > 0$, where m returns either the number of variables or the number of clauses, is known as the *exponential time hypothesis* (ETH) [24]. We have the following simple but useful proposition for single-valued measure functions.

Proposition 17. *Let X be a computational decision problem, m a complexity parameter, and ρ a single-valued polynomial-time computable and polynomially bounded measure function. Assume that the following hold:*

- *there exist sets $S_1, \ldots, S_k$ such that $\mathbb{N} = \bigcup_{i=1}^{k} S_i$,*
- *the question $s \in S_i$, $1 \leq i \leq k$ can be decided in polynomial time for integers written in unary, and*
- *there exist algorithms $A_1, \ldots, A_k$ solving $X_\rho(S_1), \ldots, X_\rho(S_k)$.*

Then, there exists a j such that A_j runs in time $\Omega(C_{X,m}^{m(I)})$.

Proof. To see this, assume A_i, $1 \leq i \leq k$, runs in time $O((C_{X,m} - \epsilon_i)^{m(I)})$ time for some $\epsilon_i > 0$, $1 \leq i \leq k$. Let I be an arbitrary instance of X. Compute $\rho(I)$ and let x be its unary representation.Then decide (in polynomial time by assumption) which set S_i that x is a member of. Finally, apply algorithm A_i to instance I. This algorithm solves X and runs in time $O((C_X - \epsilon)^{m(I)})$ where $\epsilon = \min\{\epsilon_1, \ldots, \epsilon_k\} > 0$. This obviously contradicts the choice of $C_{X,m}$. □

In particular the proposition means that we can "divide" X into finitely many subexponentially solvable subproblems if and only if X itself is solvable in subexponential time. Unfortunately, we cannot expect that there are as straightforward connections between the time complexity of a problem X and the subproblems induced via a multi-valued measure function.

Example 18. Assume that the exponential time hypothesis holds. Then, for each $k \geq 3$, the problem k-COLOURABILITY is not solvable in subexponential time when using the number of vertices in the instance as the complexity parameter [25]. We show that there exists a related CSP problem which is also not solvable in subexponential time even though there exist subexponential subproblems. Let $R_0 = \{(x, y) \in \mathbb{N}^2 \mid x \neq y\}$ and for $i > 0$ let R_i be the unary relation $\{(0), \ldots, (i+1)\}$ over $\mathbb{N}$. Let $\text{CSP}^* = \text{CSP}(\{R_0, R_1, R_2, \ldots\})$ and let ρ on $I(\text{CSP}^*)$ be the multi-valued measure function $\rho(I) = \{i \mid R_i \text{ appears in } I\}$. By letting $S_1 = \{0\}$ and $S_2 = \mathbb{N} \setminus \{0\}$ the problem CSP^*_ρ and S_1, S_2 satisfies all the properties in Proposition 17. However, first observe that CSP^* is not solvable in subexponential time since every k-COLOURABILITY instance I with n vertices can be reduced to a CSP^* instance with n variables. Both $\text{CSP}(\{R_0\})$ and $\text{CSP}(\{R_1, R_2, \ldots\})$ are however solvable in polynomial time, and are therefore subexponential. Hence, even though the subproblems are subexponential, the problem itself is not subexponential.

5.2. Constraint Satisfaction Problems and the Local-Global Conjecture

A (possibly infinite) constraint language Γ is said to be *locally tractable* if the problem $\text{CSP}(\Delta)$ is in P for all finite $\Delta \subsetneq \Gamma$. Similarly, Γ is said to be *globally tractable* if $\text{CSP}(\Gamma)$ is in P. The *local-global conjecture* states that Γ is locally tractable if and only if it is globally tractable. We say that Γ has the *local-global property* if it satisfies this conjecture. In Bodirsky & Grohe [5] it is proven that if Γ is a constraint language over a finite domain D that does not satisfy the local-global property, then there exists a constraint language Γ' over D such that $\text{CSP}(\Gamma')$ is NP-intermediate. In this section we prove a more general result not restricted to finite domains based on the notion of *extension operators*. If R is a k-ary relation and Γ a constraint language over a domain D we say that R has a *primitive positive* (p.p.) definition in Γ if

$$R(x_1, \ldots, x_k) \equiv \exists y_1, \ldots, y_l \,.\, R_1(\mathbf{x_1}) \wedge \ldots \wedge R_m(\mathbf{x_m}),$$

where each $R_i \in \Gamma \cup \{=\}$ and each $\mathbf{x_i}$ is a vector over $x_1, \ldots, x_k$, $y_1, \ldots, y_l$.

Definition 19. Let Γ be a recursively enumerable constraint language (with a suitable representation of relations in Γ). We say that $\langle \cdot \rangle$ is an *extension operator* if (1) $\langle \Gamma \rangle$ is a recursively enumerable set of p.p. definable relations over Γ, and (2) whenever $\Delta \subset \langle \Gamma \rangle$ and $\langle \Gamma \rangle \setminus \Delta$ is finite, then every $R \in \langle \Gamma \rangle \setminus \Delta$ is p.p. definable in Δ.

Another way of viewing this is that the expressive power of $\langle \Gamma \rangle$ does not change when removing finitely many relations. Since Γ and $\langle \Gamma \rangle$ are recursively enumerable we can enumerate relations in Γ or $\langle\Gamma\rangle$ as $R_1, R_2, \ldots$, and it is not hard to see that this implies that instances of $\mathrm{CSP}(\Gamma)$ and $\mathrm{CSP}(\langle\Gamma\rangle)$ are also recursively enumerable. Given an instance I of $\mathrm{CSP}(\Gamma)$ containing the relations $R_{i_1}, \ldots, R_{i_k}$, we let $\rho(I) = \{i_1, \ldots, i_k\}$. Let $\mathrm{CSP}^*_\rho(S)$ denote the $\mathrm{CSP}(\Gamma)$ problem over instances I such that $\rho(I) \subseteq S$. Define the measure function ρ' analogous to ρ but for instances over $\mathrm{CSP}(\langle\Gamma\rangle)$ using the recursive enumeration scheme for $\langle\Gamma\rangle$, and let $\mathrm{CSP}^\times_{\rho'}(S)$ be the $\mathrm{CSP}(\langle\Gamma\rangle)$ problem restricted to instances I such that $\rho'(I) \subseteq S$.

Theorem 20. *Assume Γ is a constraint language such that $\mathrm{CSP}^*_\rho(\mathbb{N})$ satisfies property P0 – P2. Let $\langle\cdot\rangle$ be an extension operator such that $\mathrm{CSP}^\times_{\rho'}(\langle\Gamma\rangle)$ satisfies property P0 – P1. If $\mathrm{P} \neq \mathrm{NP}$ then there exists a $\Gamma' \subset \langle\Gamma\rangle$ such that $\mathrm{CSP}(\Gamma')$ is NP-intermediate.*

Proof. We prove that $\mathrm{CSP}^\times_{\rho'}(\mathbb{N})$ satisfies property P0 – P3. The first two properties are trivial by assumption. For property P2 let $T = \{i_1, \ldots, i_k\}$ be an arbitrary finite subset of $\mathbb{N}$ and let $\Theta = \{R_{i_1}, \ldots, R_{i_k}\} \subseteq \langle\Gamma\rangle$. Note that Θ might contain relations which are not included in Γ. For every such relation $R \in \Theta$ we can however replace it by its p.p. definition in Γ. Let the resulting set of relations be Θ' and let $S = \{i \mid R_i \in \Theta'\}$. It is then not hard to see that $\mathrm{CSP}^\times_{\rho'}(T)$ is polynomial-time reducible to $\mathrm{CSP}^*_\rho(S)$ since every instance of $\mathrm{CSP}(\Theta')$ can be transformed to an equivalent instance of $\mathrm{CSP}(\Theta)$ by replacing every constraint application of a relation with its p.p. definition in Θ. This can be done in polynomial time since both Θ and Θ' are finite. Since $\mathrm{CSP}^*_\rho(S)$ is solvable in polynomial time by assumption it follows that $\mathrm{CSP}^\times_{\rho'}(T)$ is polynomial-time solvable, too.

For property P3 let $T \subset \mathbb{N}$ such that $\mathbb{N} \setminus T = \{t_1, \ldots, t_k\}$. To see that there exists a polynomial-time reduction from $\mathrm{CSP}^\times_{\rho'}(\mathbb{N})$ to $\mathrm{CSP}^\times_{\rho'}(T)$, we let I be an arbitrary instance of $\mathrm{CSP}^\times_{\rho'}(\mathbb{N})$. Assume I contains the constraint $R_i(x_1, \ldots, x_m)$, $i \in \mathbb{N} \setminus T$. Since $\langle\cdot\rangle$ is an extension operator the relation R_i is p.p. definable in $\langle\Gamma\rangle \setminus \Delta$ where $\Delta = \{R_i \mid i \in \mathbb{N} \setminus T\}$. Thus, we can replace $R_i(x_1, \ldots, x_m)$ with its p.p. definition in $\langle\Gamma\rangle \setminus \Delta$, and by doing this for all constraints that are not allowed by T, we end up with an instance I' of $\mathrm{CSP}^\times_{\rho'}(T)$ that is satisfiable if and only if I is satisfiable. This is a polynomial-time reduction since $\mathbb{N} \setminus T$ is a finite set.

By applying Theorem 4, we can now identify a set $S \subset \mathbb{N}$ such that $\mathrm{CSP}^\times_{\rho'}(S)$ is NP-intermediate. This implies that $\mathrm{CSP}(\Gamma')$ is NP-intermediate when $\Gamma' = \{R_i \in \langle\Gamma\rangle \mid i \in S\}$. □

Our first extension operator is based on the idea of extending a relation into a relation with higher arity. For any relation $R \subseteq D^n$, we define the *kth power* of R to be the relation

$$R^k(x_0, \ldots, x_{k \cdot n-1}) \equiv R(x_0, \ldots, x_{n-1}) \wedge R(x_n, \ldots, x_{n+n-1}) \wedge$$
$$R(x_{2n}, \ldots, x_{2n+n-1}) \wedge \ldots \wedge R(x_{(k-1)n}, \ldots, x_{(k-1)n+n-1}).$$

Given a constraint language Γ, let $\langle\Gamma\rangle_{pow} = \{R^k \mid R \in \Gamma \text{ and } k \in \mathbb{N}\}$. We represent each relation in $\langle\Gamma\rangle_{pow}$ as a pair (R, k), from which it follows that $\langle\Gamma\rangle_{pow}$ is recursively enumerable and that CSP($\langle\Gamma\rangle_{pow}$) is NP-complete if CSP($\Gamma$) is NP-complete. Now assume that $\Delta \subset \langle\Gamma\rangle_{pow}$ and that $\langle\Gamma\rangle_{pow} \setminus \Delta$ is finite. First note that for every n-ary $R' \in \langle\Gamma\rangle_{pow} \setminus \Delta$ there exists $R \in \Gamma$ and k such that $R' = R^k$. Second, since we have only removed a finite number of powers of R in Δ, there exists a sufficiently large $k' > k$ such that

$$R^k(x_1, \ldots, x_n) \equiv \exists x_{n+1}, \ldots, x_{k' \cdot n+n-1}. R^{k'+1}(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{k' \cdot n+n-1}).$$

Hence $\langle\cdot\rangle_{pow}$ is an extension operator. Extension operators are not uncommon in the literature. Well studied examples (provided relations can be suitably represented) include closure under p.p. definitions (known as *co-clones*) and closure under p.p. definitions without existential quantification (known as *partial co-clones*). These are indeed extension operators since $\langle\Gamma\rangle_{pow}$ is always a subset of the partial co-clone of Γ and hence also of the co-clone of Γ. For a general introduction to the field of clone theory we refer the reader to Lau [32].

The point of this machinery is that we now can combine the extension operator $\langle\cdot\rangle_{pow}$ with any constraint language Γ to get a problem which satisfies property P3, and to find an NP-intermediate CSP problem we only need to find a language which does not satisfy the local-global property. Let $R_{a,b,c,U} = \{(x, y) \in \mathbb{Z}^2 \mid ax - by \leq c, 0 \leq x, y \leq U\}$ for arbitrary $a, b, U \in \mathbb{N}$ and $c \in \mathbb{Z}$. Furthermore, let $\Gamma'_U = \{R_{a,b,c,U} \mid a, b \in \mathbb{N}, c \in \mathbb{Z}\}$ for any $U \in \mathbb{N}$ and the language Γ° be defined as $\Gamma^\circ = \bigcup_{i=0}^{\infty} \Gamma'_i$. Note that we can represent each relation in Γ° compactly by four integers written in binary. Due to Jonsson & Lööw [28] it is known that Γ° does not satisfy the local-global property. By combining the language Γ° and the extension operator $\langle\cdot\rangle_{pow}$ with Theorem 20 we thus obtain the following result.

Theorem 21. *If* P $\neq$ NP *then there exists a* $\Gamma' \subset \langle\Gamma^\circ\rangle_{pow}$ *such that* CSP(Γ') *is NP-intermediate.*

Due to the work of Bodirsky & Grohe [5] we already know that the CSP problem over infinite domains is non-dichotomizable. Their result is however based on reducing an already known NP-intermediate problem to a CSP problem while our language $\Gamma' \subset \langle\Gamma^\circ\rangle_{pow}$ is an explicit example of a locally tractable language obtained via blowing holes.

5.3. Locally Tractable Languages with Bounded Arity

The downside of the $\langle\cdot\rangle_{pow}$ operator is that the construction creates relations of arbitrary high arity even if the language only contain relations of bounded arity. In this section we show that simpler extensions are sometimes applicable for constraint languages over infinite domains. Assume that Γ is defined over a countably infinite domain D. For any k-ary relation R we define the $(k+1)$-ary relation R_a as $R_a(x_1, \ldots, x_n, y) \equiv R(x_1, \ldots, x_n) \wedge (y = a)$, where $a \in D$ and $(y = a)$ is the constraint application of the relation $\{(a)\}$. Let $\langle\Gamma\rangle_+ = \{R_a \mid R \in$

$\Gamma, a \in D\}$. If we represent each relation in $\langle\Gamma\rangle_+$ as a tuple (R, a) then obviously $\langle\Gamma\rangle_+$ is recursively enumerable if Γ is recursively enumerable. Now assume that Γ is an infinite constraint language and that $\langle\Gamma\rangle_+ \setminus \Delta$ is finite. For any relation $R_a \in \langle\Gamma\rangle_+ \setminus \Delta$ we first determine a b such that $R_b \in \Delta$. By construction there exists such a b since $\langle\Gamma\rangle_+ \setminus \Delta$ is finite. Then, since Γ is infinite, there exists an m-ary relation $R' \in \Gamma$ such that $R'_a \in \Delta$. Hence, we can implement R_a as

$$R_a(x_1, \ldots, x_n, y) \equiv \exists y', x'_1, \ldots, x'_m . R_b(x_1, \ldots, x_n, y') \wedge R'_a(x'_1, \ldots, x'_m, y),$$

by which it follows that $\langle\cdot\rangle_+$ is an extension operator.

Say that a language Γ is *idempotent* if for all $a \in D$ it holds that $\{(a)\}$ is p.p. definable in Γ. We assume that we can find the p.p. definition of $\{(a)\})$ in Γ in polynomial time with respect to the number of bits required to represent a.

Theorem 22. *Let Γ be an idempotent language over an infinite domain such that Γ does not satisfy the local-global property. If $\mathrm{P} \neq \mathrm{NP}$ then there exists a constraint language Γ' such that (1) $\mathrm{CSP}(\Gamma')$ is NP-intermediate and (2) Γ' contains only relations of arity $k+1$, where k is the highest arity of a relation in Γ.*

Proof. Recall that ρ is the measure function which given an instance I containing the relations $R_{i_1}, \ldots, R_{i_k}$ returns $\{i_1, \ldots, i_k\}$ (according to some enumeration of Γ), and that $\mathrm{CSP}^*_\rho(S)$ is the $\mathrm{CSP}(\Gamma)$ problem over instances I such that $\rho(I) \subseteq S$. Note also that Γ must be infinite since it does not satisfy the local-global property. Then $\mathrm{CSP}^*_\rho(\mathbb{N})$ obviously satisfies property P0–P2, and since $\langle\cdot\rangle_+$ is an extension operator, we only need to prove that $\mathrm{CSP}(\langle\Gamma\rangle_+)$ is NP-complete. NP-hardness is easy since $\mathrm{CSP}(\Gamma)$ is trivially polynomial-time reducible to $\mathrm{CSP}(\langle\Gamma\rangle_+)$. For membership in NP we give a polynomial-time reduction from $\mathrm{CSP}(\langle\Gamma\rangle_+)$ to $\mathrm{CSP}(\Gamma)$. Let I be an arbitrary instance of $\mathrm{CSP}(\langle\Gamma\rangle_+)$. For any constraint $R_a(x_1, \ldots, x_n, y)$ we replace it by $R(x_1, \ldots, x_n) \wedge \phi(x'_1, \ldots, x'_m, y)$, where $x'_1, \ldots, x'_m$ are fresh variables and where $\exists x'_1, \ldots, x'_m . \phi$ is the p.p. definition of $y = a$, which is computable in polynomial time by assumption. If we repeat the procedure for all R_a in I we get an instance I' of $\mathrm{CSP}(\Gamma)$ which is satisfiable if and only if I is satisfiable. Hence, there exists a $\Gamma' \subset \langle\Gamma\rangle_+$ such that $\mathrm{CSP}(\Gamma')$ is NP-intermediate by Theorem 20. Let k denote the highest arity of a relation in Γ. By definition every relation in $\langle\Gamma\rangle_+$ then has its arity bounded by $k+1$, which trivially also holds for Γ'. □

It is not hard to see that for the constraint language Γ° defined in the previous section any constant relation is p.p. definable in polynomial time. For any $a \in \mathbb{N}$ we simply let $(y = a) \equiv \exists x . R_{0,1,a,a}(x, y)$, i.e. the relation $0 \cdot x - 1 \cdot y \leq a \wedge 0 \leq x, y \leq a$. By Theorem 22 and the fact that Γ° only contains relations of arity 2 we therefore obtain the following.

Theorem 23. *If $\mathrm{P} \neq \mathrm{NP}$ then there exists a $\Gamma' \subset \langle\Gamma^\circ\rangle_+$ such that (1) $\mathrm{CSP}(\Gamma')$ is NP-intermediate and (2) Γ' contains only relations of arity 3.*

5.4. Propositional Abduction

Abduction is a fundamental form of nonmonotonic reasoning whose computational complexity has been thoroughly investigated [13, 16, 35]. It is known that the abduction problem parameterized with a finite constraint language is always in P, NP-complete, coNP-complete or Σ_2^{P}-complete. For infinite languages the situation differs and the question of whether it is possible to obtain a similar classification was left open in [35]. We will show that there exists an infinite constraint language such that the resulting abduction problem is NP-intermediate.

Let Γ denote a constraint language and define the propositional abduction problem $\textsc{Abd}(\Gamma)$ as follows.

INSTANCE: A tuple (V, H, M, KB), where V is a set of Boolean variables, H is a set of literals over V (known as the *set of hypotheses*), M is a literal over V (known as the the *manifestation*), and KB is a set of constraint applications $C_1(\mathbf{x}_1) \wedge \ldots \wedge C_k(\mathbf{x}_k)$ where C_i denotes an application of some relation in Γ and $\mathbf{x_i}$, $1 \leq i \leq k$, is a vector of variables in V (KB is known as the *knowledge base*).
QUESTION: Does there exist an *explanation* for I, i.e., a set $E \subseteq H$ such that $KB \wedge \bigwedge E$ is satisfiable and $KB \wedge \bigwedge E \models M$, i.e. $KB \wedge \bigwedge E \wedge \neg M$ is not satisfiable.

This simplified definition of $\textsc{Abd}(\Gamma)$ avoids some of the additional problems normally associated with abduction such as preference of minimal explanations and the class of allowed manifestations. Abduction problems based on non-classical logics with default theories have also been investigated by Eiter et. al [17]. However, the non-dichotomy result for $\textsc{Abd}(\Gamma)$ in this section also implies a non-dichotomy for the larger class of more general abduction problems.

Let Γ_{IHSB-} be the infinite constraint language consisting of the relations expressed by the *implicative hitting set-bounded* clauses $(x), (\neg x \vee y)$ and all negative clauses $\{(\neg x_1 \vee \cdots \vee \neg x_n) \mid n \geq 1\}$. We may represent each relation in Γ_{IHSB-} with a natural number in the obvious way. Let the finite constraint language $\Gamma_{IHSB-/k}$ be the subset of Γ_{IHSB-} that contains all clauses C such that $ar(C) = k$. In light of this we define the multi-valued measure function $\rho(I) = \{ar(C) \mid C \text{ is a negative clause of } KB \text{ in } I\}$. With the chosen representation of relations, ρ is obviously polynomial-time computable. We define the corresponding parameterized abduction problem $\textsc{Abd}^*_\rho(\Gamma)$ such that $I(\textsc{Abd}^*)$ is the set of abduction instances over Γ_{IHSB-}. Note the similarity between this construction and the one for the CSP problem in Section 5.2. We now verify that $\textsc{Abd}^*_\rho(\mathbb{N})$ fulfills property P0 – P3.

Property P0 holds trivially while property P1 follows from Nordh & Zanuttini [35]. For property P2, we note that if T is an arbitrary finite subset of $\mathbb{N}$, then there exists a $k \in T$ such that the clauses of every $\textsc{Abd}^*_\rho(T)$ instance is bounded by k. By [35], we know that $\textsc{Abd}(\Gamma_{IHSB-/k})$ is in P for every k, and hence that $\textsc{Abd}^*_\rho(T)$ is in P for every finite subset of S. To show property P3, we present a polynomial-time reduction from $\textsc{Abd}^*_\rho(\mathbb{N})$ to $\textsc{Abd}^*_\rho(T)$ when $\mathbb{N} \setminus T$ is finite. Let $k = \max(\mathbb{N} \setminus T)$. Arbitrarily choose an instance $I = (V, H, M, KB)$ of $\textsc{Abd}^*_\rho(\mathbb{N})$. Then, for every clause $C = (\neg x_1 \vee \ldots \vee \neg x_l) \in KB$ such that $l \in S \setminus T$, replace C by the logically equivalent clause

$$C' = (\neg x_1 \vee \ldots \vee \neg x_{l-1} \vee \neg x_l \vee \overbrace{\neg x_l \ldots \vee \neg x_l}^{k+1-l \ \neg x_l\text{'s}})$$

of length $k + 1$. If we let the resulting knowledge base be KB' then $I' = (V, H, M, KB')$ is an instance of $\textsc{Abd}^*_\rho(T)$ which has a solution if and only if I has a solution.

From this and Theorem 4 it follows that there exists a $S' \subset \mathbb{N}$ such that $\textsc{Abd}^*_\rho(S')$ is NP-intermediate. Hence, we conclude the following.

Theorem 24. *If* P $\neq$ NP *then there exists a constraint language* $\Gamma'_{IHSB-} \subset \Gamma_{IHSB-}$ *such that* $\textsc{Abd}(\Gamma'_{IHSB-})$ *is NP-intermediate.*

6. Future Work

One way of obtaining genuinely new NP-intermediate problems is to consider other complexity-theoretic assumptions than P $\neq$ NP. We have pointed out that the LogClique problem is NP-intermediate under the ETH, and that the main difficulty is to provide a lower bound, i.e. proving that LogClique $\notin$ P. One may suspect that providing lower bounds is the main difficulty also when considering other problems. We have seen that CSP problems constitute a rich source of NP-intermediate problems via different kinds of parameterizations, Hence, it appears feasible that methods for studying the complexity of parameterized problems will become highly relevant. In particular, *linear fpt-reductions* [10, 11] have been used for proving particularly strong lower bounds which may be used for linking together NP-intermediate problems, parameterized problems, and lower bound assumptions.

The structure of the set of NP-intermediate problems under various reductions has been studied to some extent, cf. Ambos-Spies [1], Ladner [30], Landweber et al. [31], and Schöning [39]. We suggest that these kind of questions may be studied via *clone theory.* Clone theory is a well-studied subarea of universal algebra which has proven to be very powerful when studying the computational complexity of satisfiability and constraint satisfaction problems [8, 26]. Additionally, it has recently been shown to be useful for studying bounds on the time complexity of NP-complete problems [27]. The results in [27] are based on studying a particular kind of clones (known as *strong partial clones*) with the aim of obtaining reductions that increases size as little as possible. Such reductions provide a fine-grained picture of the time complexity of problem classes, which may be useful when studying NP-intermediate problems. In particular a Boolean relation R is proven to have the property that $\mathrm{CSP}(\{R\})$ is solvable at least as fast as any other Boolean $\mathrm{CSP}(\cdot)$ problem, with respect to worst case $O(c^n)$ complexity [27]. Attempting to restrict this problem even further with structural restrictions could in the best case open up new sources of NP-intermediate problems. We have also noted that recent results by Dell and van

Melkebeek [15] can be used for proving the non-existence of such fine-grained reductions under the assumption that the polynomial-time hierarchy does not collapse. This opens up new ways for studying the class of NP-intermediate problems.

We have shown that the propositional abduction problem has NP-intermediate fragments. One may view abduction as a problem that is closely related to Boolean CSPs. However, there is an important difference: the CSP(Γ) problem is either a member of P or NP-complete for *all* choices of Boolean Γ. Hence, it would be interesting to determine which finite-domain CSP-related problems can be used for obtaining NP-intermediate problems and which of them have the local-global property. Inspired by our result on the abduction problem, we view other forms of non-monotonic reasoning such as circumscription and default logic as potential candidates. Unfortunately, many problems of this type are polynomial-time solvable only in very restricted cases, For example, the minimal constraint inference problem for propositional circumscription (MIN-INF-CSP(Γ)) is coNP-hard even in extremely restricted cases [9] such as when Γ only contains the OR relation $\{(0,1),(1,0),(1,1)\}$. This makes it hard to find candidate languages which does not satisfy the local-global property. Thus, more powerful methods than blowing holes may be needed for identifying NP-intermediate problems in this and similar cases.

Acknowledgements

Peter Jonsson is partially supported by Swedish Research Council (VR) under grant 621-2012-3239. Victor Lagerkvist is supported by the National Graduate School in Computer Science (CUGS), Sweden.

References

[1] K. Ambos-Spies. Sublattices of the polynomial time degrees. *Information and Control*, 65(1):63–84, 1985.

[2] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Matrix Analysis and Applications*, 8(2):277–284, 1987.

[3] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[4] M. Bodirsky. Complexity classification in infinite-domain constraint satisfaction. Habilitation thesis. Hannover, Germany, Univ. Paris 7., 2012.

[5] M. Bodirsky and M. Grohe. Non-dichotomies in constraint satisfaction complexity. In *Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP-2008)*, pages 184–196, 2008.

[6] H. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[7] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, January 2006.

[8] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the computational complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720 – 742, 2005.

[9] M. Cadoli and M. Lenzerini. The complexity of closed world reasoning and circumscription. *Journal of Computer and System Sciences*, 48:255–301, 1994.

[10] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proc. 19th IEEE Conference on Computational Complexity (CCC-2004)*, pages 150–160, 2004.

[11] J. Chen, X. Huang, I. Kanj, and G. Xia. Linear fpt reductions and computational lower bounds. In *Proc. 36th ACM Symposium on Theory of Computing (STOC-2004)*, pages 212–221, 2004.

[12] Y. Chen, M. Thurley, and M. Weyer. Understanding the complexity of induced subgraph isomorphisms. In *Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP-2008)*, pages 587–596, 2008.

[13] N. Creignou, J. Schmidt, and M. Thomas. Complexity of propositional abduction for restricted sets of Boolean functions. In *Proc. 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR-2010)*, 2010.

[14] R. Dechter. *Constraint processing.* Elsevier Morgan Kaufmann, 2003.

[15] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC-2010)*, pages 251–260, 2010.

[16] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):3–42, 1995.

[17] T. Eiter, G. Gottlob, and N. Leone. Semantics and complexity of abduction from default theories. *Artificial Intelligence*, 90(1-2):177 – 223, 1997.

[18] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[19] M. Garey and D. Johnson. "Strong" NP-completeness results: motivation, examples and implications. *Journal of the ACM*, 25(3):499–508, 1978.

[20] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman and Company, 1979.

[21] C. Glaßer, A. Selman, S. Travers, and K. Wagner. The complexity of unions of disjoint sets. *Journal of Computer and System Sciences*, 74(7):1173–1187, 2008.

[22] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.

[23] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2006)*, pages 289 – 298, 2006.

[24] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[25] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[26] P. Jeavons, D. A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[27] P. Jonsson, V. Lagerkvist, G. Nordh, and B. Zanuttini. Complexity of SAT problems, clone theory and the exponential time hypothesis. In *Proc. the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 1264–1277, 2013.

[28] P. Jonsson and T. Lööw. Computational complexity of linear constraints over the integers. *Artificial Intelligence*, 195:44 – 62, 2013.

[29] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[30] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[31] L. H. Landweber, R. J. Lipton, and E. L. Robertson. On the structure of sets in NP and other complexity classes. *Theoretical Computer Science*, 15:181–200, 1981.

[32] D. Lau. *Function Algebras on Finite Sets: Basic Course on Many-Valued Logic and Clone Theory (Springer Monographs in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[33] D. Marx. Approximating fractional hypertree width. *ACM Transactions on Algorithms*, 6(2), 2010.

[34] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC-2010)*, pages 735–744, 2010.

[35] G. Nordh and B. Zanuttini. What makes propositional abduction tractable. *Artificial Intelligence*, 172:1245–1284, 2008.

[36] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.

[37] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[38] T. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)*, pages 216–226. ACM Press, 1978.

[39] U. Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theoretical Computer Science*, 18:95–103, 1982.

[40] A. Selman. Natural self-reducible sets. *SIAM Journal on Computing*, 17(5):989–996, 1988.