Parameterized Complexity of CSP for Infinite Constraint Languages

Ruhollah Majdoddin
Humboldt Universität zu Berlin, Germany
r.majdodin@gmail.com

We study parameterized Constraint Satisfaction Problem for infinite constraint languages. The parameters that we study are weight of the satisfying assignment, number of constraints, maximum number of occurrences of a variable in the instance, and maximum number of occurrences of a variable in each constraint. A dichotomy theorem is already known for finite constraint languages with the weight parameter. We prove some general theorems that show, as new results, that some well-known problems are fixed-parameter tractable and some others are in \( W[1] \).

1 Introduction

A constraint language is a domain and a set of relations over this domain. We study the parameterized complexity of Constraint Satisfaction Problem (CSP) for infinite Boolean constraint languages (where each relation has a finite arity). This is a new subject, as it seems that the works which explicitly address the (parameterized) complexity of CSP have been concerned with finite constraint languages.

The parameters that we study are \( k \), weight of a satisfying assignment, \( k \leq \), the maximum weight of a satisfying assignment, the number \( u \) of constraints, the maximum number \( t \) of occurrences of a variable in the instance, and the maximum number \( e \) of occurrences of a variable in a constraint. Marx \cite{Marx} proves a dichotomy theorem for CSP with parameter \( k \) for finite constraint languages over the Boolean domain. This is extended by Marx and Bulatov \cite{MarxBulatov} to all finite domains. Letting the constraint language to be infinite makes the problem not just much more general, but also much more harder. Because, for example, many proves in \cite{Marx} and \cite{MarxBulatov} use the fact that there is a bound on the arity of relations in a finite constraint language, but there is no such bound for an infinite constraint language.

We study constraint languages that are fpt-membership checkable, that is there is an fpt-algorithm that given the index of a relation and a tuple, the algorithm decides whether the tuple is in the relation (the parameter is the weight of the tuple, that is the number of 1s in the tuple).

Our mathematical concepts and notation are described in detail in the next section. Many of our results are about constraint languages \( W^A \) for some set \( A \subseteq \mathbb{N}_0 \). Roughly speaking, \( W^A \) has symmetric relations of every arity, where \( A \) is the set of permitted weights of the tuples in the relations. For an integer \( c \geq 0 \), \( W^c \) is the union of all \( W^A \) for any \( A \subseteq [0, c] \).

We have two groups of results. In the first group, sections 3 and 4, we study CSP with additional parameters besides \( k \). First, we prove that for every set \( E \subseteq \mathbb{N}_0 \), the problem CSP(\( W^E \))\(_{k,u,e} \) is fixed-parameter tractable. Moreover, if \( E \) or \( \mathbb{N}_0 \setminus E \) is finite, then CSP(\( W^E \))\(_{k,u} \) is fixed-parameter tractable. Notice that a trivial bounded search tree method does not give an fpt-algorithm here (See e.g. \cite{Iwama} Sec. 1.3), because \( W^E \) is an infinite constraint language and there is no bound on the arity of the its relations. Moreover, the additional parameter \( u \) is necessary, because CSP(\( W^{\{1\}} \))\(_{k} \), called Weighted Exact CNF, is \( W[1] \)-hard \cite{Iwama}.
It then follows that for every set $E \subseteq \mathbb{N}$, the problem $\text{CSP}(W^E)_{k,t,e}$ is fixed-parameter tractable. Moreover, if $E$ or $\mathbb{N} \setminus E$ is finite, then $\text{CSP}(W^E)_{k,t}$ is fixed-parameter tractable. In the following we present some interesting examples.

- $\text{CSP}(W^0)_{k,u}$ and $\text{CSP}(W^1)_{k,t}$ are fixed-parameter tractable.

Notice that $\text{CSP}(W^q)_{k}$ is equivalent to $p\text{-WSAT}(\text{CNF}^+)$). This result is interesting, because $p\text{-WSAT}(\text{CNF}^+)$ and $p\text{-WSAT}(\text{CNF})$ are both $W[2]$-complete [10]. It is noteworthy that for every $d \geq 1$, $p\text{-WSAT}(d\text{-CNF})$ and $p\text{-WSAT}(d\text{-CNF}^+)$ are $W[1]$-complete [10], but $p\text{-WSAT}(d\text{-CNF}^+)$ is fixed-parameter tractable [13]. Finally $p\text{-clausesize-WSAT}(\text{CNF})$ is $W[1]$-complete.

- $\text{CSP}(W^{(1)})_{k,u}$ and $\text{CSP}(W^{(1)})_{k,t}$ are fixed-parameter tractable.

This is the problem Weighted Exact CNF with the additional parameter $u$ or $t$.

- $\text{CSP}(W^{\text{odd}})_{k,t \leq 3}$ and $\text{CSP}(W^{\text{even}})_{k \leq u}$ are fixed-parameter tractable.


It is proved in [9] that $\text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k \leq u}$, $\text{CSP}(W^{\text{odd}})_{k \leq u}$ and even $\text{CSP}(W^{\text{even}})_{k}$ are $W[1]$-hard. Arvind et al. [11] proved that the hardness holds also with the additional parameter $t$, and even if $t$ is bounded to $t \leq 3$, that is $\text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k \leq u, t \leq 3}$ and $\text{CSP}(W^{\text{even}})_{k,t \leq 3}$ are $W[1]$-hard. Notice that both papers study these problems in the setting of linear equations $Ax = b$ over $\mathbb{F}_2$. But, to the best of our knowledge, the complexity of $\text{CSP}(W^{\text{odd}})_{k,t \leq 3}$ has been left open, and by our result above, it is fixed-parameter tractable. We find this somewhat surprising, as it is the only known case of parameterized CSP over parity constraint languages that introducing the additional parameter $t$ reduces the complexity of the problem.

An important open problem is whether $\text{CSP}(W^{\text{odd}})_{k \leq u}$, called Even Set, is $W[1]$-hard [6]. It is even not known whether $\text{CSP}(W^{\text{even}})_{k \leq u}$ is $W[1]$-hard. Still, we prove that $\text{CSP}(W^{\text{even}})_{k \leq u}$ is fixed-parameter tractable. Also, it has been left open whether $\text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,t}$ is in $W[1]$. Our second result answers this positively: For every (possibly infinite) constraint language $\Gamma$, it holds $\text{CSP}(\Gamma)_{k,t} \in W[1]$.

Our second group of results, presented in sections 5 and 6, is about containment in $W[1]$. Downey and Fellows showed that that Weighted Exact CNF, in our setting $\text{CSP}(W^{(1)})_{k}$, reduces to $p$-Perfect Code and vice versa, and proved that the problem is $W[1]$-hard by means of a reduction from $p$-Independent-Set [7][8]. They conjectured that the problem could be of difficulty intermediate between $W[1]$ and $W[2]$, and thus not $W[1]$-complete [8] pp. 277, 458, 487. Surprisingly, Cesati proved that $\text{CSP}(W^{(1)})_{k} \in W[1][4][5]$. He uses reductions to Short NonDeterministic Turing Machine Acceptance. In this problem the Turing Machine can have fpt-size state space and alphabet, but the parameter is the runtime of the machine.

A natural question is whether this result can be generalized. First, by a somewhat technical proof, which is an adaption of the proof of [4], we show that for every $d \geq 0$, $\text{CSP}(CW^d)_{k} \in W[1]$. Notice that $W^d \subset CW^d$. We generalize this, by still another involved proof, as follows: If for a (possibly infinite) constraint language $\Gamma$, there be an integer $d \geq 1$ such that for every relation $R \in \Gamma$, size of each set $T \in R$ is at most $d$, then $\text{CSP}(\Gamma)_{k} \in W[1]$. This implies our ultimate generalization: For every $d \geq 0$, $\text{CSP}(W^d)_{k} \in W[1]$.

## 2 Preliminaries

We denote the set of positive integers by $\mathbb{N}$ and the set of nonnegative integers by $\mathbb{N}_0$. For integers $0 \leq a \leq b$, we use the notation $[a,b] := \{a, a+1, \ldots, b\}$. For an integer $a > 0$, we write $[a] := [1, \ldots, a]$, and we denote $[0] := \emptyset$. We encode integers in binary.

We use the abbreviations

\[
\begin{align*}
\text{even} & := \{i | i = 2j, j \in \mathbb{N}_0\}, \\
\text{odd} & := \{i | i = 2j + 1, j \in \mathbb{N}_0\}.
\end{align*}
\]

(1)

For a set $X$, we denote the powerset of $X$ by $2^X$. For sets $A$ and $B$ and a function $f : A \rightarrow B$, the image of a set $D \subseteq A$ under $f$ is defined as $f(D) := \{y \in B | \exists x \in A, y = f(x)\}$, and the preimage of a set $E \subseteq B$ under $f$ is defined as $f^{-1}(E) := \{x \in A | f(x) \in E\}$.
We use the basic definitions of parameterized complexity theory in [11], including the following. Let $\Sigma$ be a nonempty finite alphabet. We refer to sets $Q \subseteq \Sigma^*$ of strings over $\Sigma$ as classical decision problem. A parameterization of $\Sigma^*$ is a mapping $\kappa : \Sigma^* \to \mathbb{N}$ that is polynomial time computable. A parameterized problem (over $\Sigma$) is a pair $(Q, \kappa)$ consisting of a set $Q \subseteq \Sigma^*$ of strings over $\Sigma$ and a parameterization $\kappa$ of $\Sigma^*$.

Let $D$ be a set. A relation $R$ of arity $\text{arity}(R) \geq 1$ over domain $D$ is a subset: $R \subseteq D^{\text{arity}(R)}$. In this paper we consider only relations of finite arity. A set $\Gamma$ of relations over $D$, $\Gamma := \{R_i\}_{i \in \mathbb{N}}$, is called a constraint language over domain $D$. For a constraint $R_i \in \Gamma$, $i$ is called the index of $R_i$ (in $\Gamma$).

In this paper, other than in the context of structures and first order logic (Section 3), we always have Boolean domains. That is $D = \{0, 1\}$, and we identify a tuple in $D^r$ with a subset of $[r]$, which is the set of 1 positions of the tuple. This results in the definition of a Boolean relation as a set of subsets: $R \subseteq 2^{\text{arity}(R)}$. This is the definition that we use in this paper. A Boolean constraint language is defined likewise.

A constraint language $\Gamma := \{R_i\}_{i \in \mathbb{N}}$ is called membership checkable if there is a Turing machine $B$ (membership checker) such that for any relation $R_i \in \Gamma$ of arity say $q$, and for any $T \subseteq [q]$, the machine $B$ with input $(i, T)$ outputs 1 if $T \in R_i$, and 0 otherwise. We say $\Gamma$ is fpt membership checkable if there is a computable function $f_B : N_0 \to \{0, 1\}$ and an integer $c$ such that $B$ halts in at most $f_B(|T|) (\log i)^c$ steps. We assume that all the constraint languages in this paper are fpt membership checkable. This is arguably a mild condition and includes many interesting natural problems.

For a set $A \subseteq N_0$, we define the relation $W^A_m$ of arity $m$ as

$$W^A_m := \{T \subseteq [m], |T| \in A\},$$

and the constraint language $W^A = \{W^A_i\}_{i \in \mathbb{N}}$. The constraint languages $W^{\text{odd}}$ and $W^{\text{even}}$ are defined accordingly. For an integer $c \geq 0$, define the constraint language $W^c$ as

$$W^c := \bigcup_{A \subseteq [0, c]} W^A.$$  \hspace{2in} (3)

Define the constraint language $W$ as

$$W := \bigcup_{A \subseteq N_0} W^A.$$  \hspace{2in} (4)

One interesting example is $W^{|n|}$, which is equivalent to disjunction. So $W^N$ is the constraint language of disjunctions of any arity. The other interesting example is $W^{\{1\}}$, which can be used to define the WEIGHTED EXACT CNF problem.

For a set $A \subseteq N_0$ and integers $d, m \geq 0$, we define the relation conditional weight, $CW^A_{d,m}$, of arity $d + m$, as

$$CW^A_{d,m} := \{T \subseteq [d + m] \mid |d| \subseteq T \Rightarrow |d + 1, d + m] \cap T| \in A\},$$

and the constraint language

$$CW^A := \{CW^A_{d,m}\}_{d,m \geq 0}.$$  \hspace{2in} (6)

A degenerate case is $d = 0$, where

$$CW^A_{0,m} := W^A_m.$$  \hspace{2in} (7)

Another degenerate case is $0 \notin A$ and $m = 0$, then $CW^A_{d,0}$ is equivalent to Nand. A Horn clause can be expressed as $CW^A_{d,1}$.

For a constraint language $\Gamma = \{R_i\}_{i \in \mathbb{N}}$ and a set of variable symbols (or simply variables) $V$, a constraint $\psi$ is a pair $\psi := (m, (v_1, \ldots, v_r))$, where $m \in \mathbb{N}$, $r := \text{arity}(R_m)$ and $(v_1, \ldots, v_r) \in V^r$. We
denote ψ either by \( R_m(v_1, \ldots, v_r) \), or by \( R_m(\omega) \), for the mapping \( \omega : [r] \to V \), where for every \( i \in [r] \), \( \omega(i) := v_i \).

An assignment \( D \) of \( V \) is a subset of \( V \). We say that \( D \) satisfies \( \psi \) if \( T \in R_m \), where \( T := \{ i | i \in [r], v_i \in D \} \).

We denote CSP for a constraint language \( \Gamma \) by CSP(\( \Gamma \)). An instance \( I \) of CSP(\( \Gamma \)) is a pair \( I := (V,C) \), where \( V \) is the set of variables and \( C \) is a sequence of constraints for \( \Gamma \). We write \( |C| \) to denote the length of \( C \), and write \( C(i) \) to denote the \( i \)th constraint in \( C \). An assignment \( D \) of \( V \) satisfies \( I \), if \( D \) satisfies every constraint in \( C \).

Parameterized CSP has exactly one of the parameters \( k \) and \( k_\leq \), and possibly any of the parameters \( t, u \) and \( e \).

Parameter \( k \) is called weight, and an instance of CSP with this parameter, CSP(\( \Gamma \))\( _k \), is a pair \( I := (V,C) \), where \( C \) is defined as in CSP(\( \Gamma \)), but additionally the first constraint is of form \( C(1) := W_{|V|}^{(k_0)}(v_1, \ldots, v_m) \), where \( V := \{ v_1, \ldots, v_m \} \). We define \( k(I) := k_0 \). This means that if some \( D \subseteq V \) is a satisfying assignment, then \( |D| = k_0 \).

Likewise, parameter \( k_\leq \) is called weight-less-than, and an instance of CSP with this parameter, CSP(\( \Gamma \))\( _{k_\leq} \), is a pair \( I := (V,C) \), where \( C \) is defined as in CSP(\( \Gamma \)), but additionally the first constraint is of form \( C(1) := W_{|V|}^{(k_0)}(v_1, \ldots, v_m) \), where \( V := \{ v_1, \ldots, v_m \} \). We define \( k_{\leq}(I) := k_0 \). This means that if some \( D \subseteq V \) is a satisfying assignment, then \( |D| \leq k_0 \).

It was possible, and is common in the literature, to give the (integer) value of the parameters \( k \) and \( k_\leq \) as part of the input. But our definition of the parameterized problem is arguably more homogeneous, as it demonstrates that these weight conditions are indeed just another constraint.

For an instance \( I := (V,C) \) of CSP(\( \Gamma \))\( _{k,u,t,e} \), \( u \) is the number of constraints,

\[
u(I) := |C|,
\]

\( t \) is the maximum number of occurrences of any variable in the whole instance,

\[
t(I) := \max_{v \in V} \sum_{\substack{i\in|[C]| \\ C(i) = R(\omega) \\ v \in i}} |\omega^{-1}(\{v\})|,
\]

and \( e \) is the maximum number of occurrences of any variable in any constraint,

\[
e(I) := \max_{v \in V} \max_{\substack{i\in|[C]| \\ C(i) = R(\omega) \\ \exists R \Gamma_2}} |\omega^{-1}(\{v\})|.
\]

We write CSP(\( \Gamma_1 \))\( _k \Gamma_2 \)\( _{k_\leq} \) to denote parameterized CSP for the constraint language \( \Gamma_1 \cup \Gamma_2 \), where parameter \( k \) is defined as usual, but the parameter \( t \) applies only to the constraints for \( \Gamma_1 \). More formally, for an instance \( I := (V,C) \)

\[
t(I) := \max_{v \in V} \sum_{\substack{i\in|[C]| \\ C(i) = R(\omega) \\ \exists R \Gamma_2 \}} |\omega^{-1}(v)|.
\]

Sometimes the problems are restricted to those instances with a fixed or bounded parameter value. We denote this in the parameter list. For example, for each instance \( I \) of problem CSP(\( \Gamma \))\( _{k,t \leq 3} \), it holds \( t(I) \leq 3 \).

**Fact 2.1.** For every constraint languages \( \Gamma \), CSP(\( \Gamma \))\( _{k_\leq} \) \( \leq_{fpt} \) CSP(\( \Gamma \))\( _k \).

We use the notation of [11] for propositional logic, including the following. We denote by CNF the class of all propositional formulas in *conjunctive normal form*. CNF\( ^+ \) denotes the subclass of CNF that there is no negation symbol in a formula, and CNF\( ^- \) denotes the subclass of CNF that there is a negation symbol in front of every variable in a formula.
For any class \( A \) of propositional formulas, the parameterized weighted satisfiability problem for \( A \) is defined as follows:

\[
p\text{-WSat}(A)
\]

**Instance:** \( \alpha \in A \) and \( k \in \mathbb{N} \).

**Parameter:** \( k \).

**Question:** Decide whether \( \alpha \) is \( k \)-satisfiable.

For class CNF we also define

\[
p\text{-clause-size-WSat}(\text{CNF})
\]

**Instance:** \( \alpha \in \text{CNF} \) and \( k \in \mathbb{N} \).

**Parameter:** \( k + d \), where \( d \) is the maximum number of literals in a clause of \( \alpha \).

**Question:** Decide whether \( \alpha \) is \( k \)-satisfiable.

We denote the size of input by \( n \).

### 3 Number of Occurrences of Variables - in FPT

We use the notation for first-order logic from [11]. The class of all first order formulas is denoted by \( \text{FO} \).

A (relational) vocabulary \( \tau \) is a set of relation symbols. Each relation symbol \( R \) has an *arity* \( \text{arity}(R) \geq 1 \). A *structure* \( A \) of vocabulary \( \tau \), or \( \tau \)-*structure* (or simply *structure*), consists of a set \( A \) called the *universe* and and interpretation \( R^A \subseteq A^{\text{arity}(R)} \) of each relation symbol \( R \in \tau \). We write \( R^A \) to denote that the tuple \( \bar{a} \in A^{\text{arity}(R)} \) belongs to the relation \( R^A \).

The parameterized model-checking problem for a class \( \Phi \) of formulas is defined as follows.

\[
p\text{-MC}(\Phi)
\]

**Instance:** A structure \( A \) and a formula \( \varphi \in \Phi \).

**Parameter:** \( |\varphi| \).

**Question:** Decide whether \( \varphi(A) \not= \emptyset \).

The restriction of \( p\text{-MC}(\Phi) \) to input structures from a class \( D \) of structures is denoted by \( p\text{-MC}(D, \Phi) \).

Let \( A \) be a \( \tau \)-structure. The *Gaifman graph* (or primal graph) of \( \tau \)-structure \( A \) is the graph \( G(A):= (V, E) \), where \( V := A \) and

\[
E := \{(a, b)|a, b \in A, a \neq b, \text{ there exists an } R \in \tau \text{ and a tuple } (a_1, \ldots, a_r) \in R^A, \text{ where } r := \text{arity}(R), \text{such that } a, b \in \{a_1, \ldots, a_r\}\}
\]

The *degree* of a structure \( A \) is the degree of its Gaifman graph \( G(A) \).

The following corollary follows from [12].

**Theorem 3.1** (see [11 Corollary 12.23]). Let \( d \in \mathbb{N} \). The parameterized model-checking problem for first-order logic on the class of structures of degree at most \( d \) is fixed-parameter tractable.

**Theorem 3.2.** Let \( E \subseteq \mathbb{N}_0 \). Then \( \text{CSP}(W^E_{k,u,c}) \) is fixed-parameter tractable. Moreover, if \( E \) or \( \mathbb{N}_0 \setminus E \) is finite, then \( \text{CSP}(W^E_{k,u}) \) is fixed-parameter tractable.

**Proof.** To each instance \( I = (V, C) \) corresponds a structure \( A \) and a formula \( \varphi \in \text{FO} \), described below, such that \( \varphi(A) \not= \emptyset \) if and only if \( I \) has a solution.

Let \( h := \min\{e(I), \max E, \max(\mathbb{N}_0 \setminus E)\} \). Let the universe of \( A \) be \( A := V \), and \( \tau := \{R_{i,j} | i \in [2, |C|], j \in [0, h]\} \), where each \( R_{i,j} \) is a unary relation as follows. Let the \( C(i) := W^E_d(\sigma) \). Then \( \forall v \in V \ R^A_{i,j}v \) if and only if \( |\sigma^{-1}(v)| = j \). Because the vocabulary \( \tau \) has only unary relations, the degree of structure \( A \) is 0. Let \( D \) be the class of all structures constructed this way. It follows from Theorem 3.1 that \( p\text{-MC}(D, \text{FO}) \) is fixed-parameter tractable.
Let \( k_0 := k(I) \). Let formulas \( \varphi_1 \) and \( \varphi_2 \) be as follows

\[
\varphi_1 := \exists x_1 \ldots \exists x_{k_0} \left( \bigwedge_{i,j \in [k_0], i \neq j} x_i \neq x_j \right) \wedge \bigwedge_{i \in [2,|C|]} \bigvee_{c_1, \ldots, c_{k_0} \in [0, h]} \bigwedge_{j \in [k_0]} R^A_{i,c,j} x_j, \tag{13}
\]

\[
\varphi_2 := \exists x_1 \ldots \exists x_{k_0} \left( \bigwedge_{i,j \in [k_0], i \neq j} x_i \neq x_j \right) \wedge \bigwedge_{i \in [2,|C|]} \bigwedge_{c_1, \ldots, c_{k_0} \in [0, h]} \bigwedge_{j \in [k_0]} \neg R^A_{i,c,j} x_j. \tag{14}
\]

If \( E \) is finite, set \( \varphi := \varphi_1 \). If \( \mathbb{N}_0 \setminus E \) is finite, set \( \varphi := \varphi_2 \). If \( e(I) \) is finite, then \( \varphi \) can be set as \( \varphi_1 \) or \( \varphi_2 \). It is straightforward to see that \( |\varphi| \) depends only on \( k_0, h \) and \( u(I) \). This completes the proof. \( \square \)

Notice that by Fact. 2.1, the above result still holds if parameter \( k \) is replaced by \( k_{\leq} \).

**Fact 3.3.** \( \text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,u} \) and \( \text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,t,e} \) are fpt-reducible to \( \text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,u} \) and \( \text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,t} \), respectively. The reduction can be computed in polynomial time.

By the above fact, we get the following corollary.

**Corollary 3.4.** \( \text{CSP}(W^N)_{k,u} \), \( \text{CSP}(W^{(1)})_{k,u} \) and \( \text{CSP}(W^{\text{odd}} \cup W^{\text{even}})_{k,u} \) are fixed-parameter tractable.

Observing in Theorem 3.3 that if \( 0 \notin E \) and \( u(I) > t(I)k(I) + 1 \), then the instance has no satisfying assignment, and with Fact 3.3 we get the following corollary about when \( k \) and \( t \) are the parameters.

**Corollary 3.5.** Let \( E \subseteq \mathbb{N} \) (so \( 0 \notin E \)). Then \( \text{CSP}(W^E)_{k,t,e} \) is fixed-parameter tractable. Moreover, if \( E \) or \( \mathbb{N} \setminus E \) is finite, then \( \text{CSP}(W^E)_{k,t} \) is fixed-parameter tractable. Especially \( \text{CSP}(W^N)_{k,t}, \text{CSP}(W^{(1)})_{k,t} \), \( \text{CSP}(W^{\text{odd}})_{k,t} \) are fixed-parameter tractable.

### 4 Number of Occurrences of Variables - in \( W[1] \)

The following problem is \( W[1] \)-complete [13, 8].

**Short NonDeterministic Turing Machine Acceptance**

**Instance:** A single-tape, single-head nondeterministic Turing machine \( M \); a word \( x \) over the alphabet of \( M \); a positive integer \( l \in \mathbb{N} \).

**Parameter:** \( l \).

**Question:** Is there a computation of \( M \) on input \( x \) that reaches a final accepting state in at most \( l \) steps?

**Theorem 4.1.** Let \( \Gamma \) be a (possibly infinite) constraint language. Then \( \text{CSP}(\Gamma)_{k,t} \in W[1] \).

**Proof.** Let \( \Gamma := \{ R_i \}_{i \in \mathbb{N}} \) and the Turing machine \( B \) be the fpt membership checker of \( \Gamma \). We present an fpt-reduction that maps any given instance \( I = (V,C) \) (with the parameter values \( k(I) = k_0 \) and \( t(I) = t_0 \)) to an instance \( J \) of Short NonDeterministic Turing Machine Acceptance. Let \( m := |C| \), and the constraints from \( \Gamma \) in \( C \) be \( R_{h_i}(\omega_i) \) for \( i \in [2, m] \).

For a variable \( v \in V \), define the set \( E_v \) as

\[
E_v := \{ i | i \in [m], \omega_i^{-1}\{v\} \neq \emptyset \}. \tag{15}
\]
Notice that \(|E_v| \leq t_0\). Let \(D\) be the set
\[
D := \{i| i \in [m], \emptyset \not\in R_h_i}\.
\]

If \(|D| > k_0 \cdot t_0\), then the machine \(M\) of instance \(J\) simply rejects. Otherwise \(M\), knowing the set \(D\) and also the sets \(E_v\) for any variable \(v \in V\), starts with an empty tape and runs in three steps.

**Step 1.** \(M\) chooses nondeterministically the set of variables \(A \subseteq V\) of size \(|A| \leq k_0\).

**Step 2.** For each \(i \in [2, m]\) such that \(i \in E_v\) for some \(v \in A\), \(M\) computes the set \(T := \{j | \omega_i(j) \in A\}\). \(M\) runs \(B(h_i, T)\), and if the output is 0, then \(M\) rejects.

**Step 3.** If \(D \not\subseteq \bigcup_{v \in A} E_v\), then \(M\) rejects. Otherwise \(M\) accepts.

We should now show that machine \(M\) can indeed run machine \(B\) efficiently. This assures that parameter \(l\) of instance \(J\), that is runtime of \(M\), is a fixed function of \(k_0\) and \(t_0\). We will use the following basic theorems.

**Linear Speedup Theorem (see [15](Theorem 2.2))** - Let \(L \in \text{TIME}(g(n))\). Then, for any \(\epsilon > 0\), \(L \in \text{TIME}(g'(n))\), where \(g'(n) = cg(n) + n + 2\).

In the proof of Linear Speedup Theorem, if \(G\) is the machine deciding \(L\) with runtime \(g(n)\) and \(G\) has \(d\) tapes and alphabet \(\Sigma\), then a machine \(G'\) with runtime \(g'(n)\) is constructed, such that \(G'\) has \(d\) tapes if \(d > 2\) and \(2\) tapes if \(d \leq 2\), and its alphabet is \(\Sigma' = \Sigma \cup \Sigma^{(\log i)}\). The cause of the \(n + 2\) part of \(g'(n)\) is that \(G'\) should scan the whole input and translate it in its own alphabet.

**1-tape Simulation Theorem (see [15](Theorem 2.2))** - Given any \(t\)-tape Turing machine \(H\) operating within time \(f(n)\), we can construct a 1-tape Turing machine \(H'\) operating within time \(O(f(n)^t)\) and such that, for any input \(x\), \(H(x) = H'(x)\).

In the proof of the above theorem, if \(H\) has alphabet \(\Sigma\), then the size of the alphabet of \(H'\) is \(2|\Sigma| + 2\).

Remember that runtime of \(B\) on input \((i, T)\) is, by definition, \(f_B([|T|])/(\log i)^\epsilon\) for some fixed \(c\). Now let \(\Sigma\) be the alphabet of \(B\). Then by applying the Linear Speedup Theorem for \(\epsilon = (1/(\log i)^\epsilon)\), there is a machine \(B'\) that has two tapes and an alphabet \(\Sigma'\) of size \(|\Sigma'| = |\Sigma| + |\Sigma|^{(\log i)^\epsilon}\) and has runtime \(f_B([|T|]) + O((\log i + |T| + 2))\), such that \(B'\) and \(B\) have the same output on all inputs. Therefore, by 1-tape Simulation Theorem above, machine \(M\) having one tape and an alphabet of size at least \(2|\Sigma'| + 2\) can simulate \(B'\) in time \(O((f_B([|T|]))^4)\). The \(O((\log i + |T| + 2))\) part of the runtime can be avoided because \(B'\) simulated by \(M\) does not have to translate the input to its alphabet.

**Corollary 4.2.** \(\text{CSP}(W_{\text{odd}} \cup W_{\text{even}})_{k, t} \in W[1]\).

## 5 CW Formulas

**Theorem 5.1.** Let \(b \geq 0\). Then \(\text{CSP}(\text{CW}^b)_{k}\) is in \(W[1]\).

**Proof.** We present an fpt-reduction from \(\text{CSP}(\text{CW}^b)_{k}\) to \(\text{ShortNonDeterministic Turing Machine Acceptance.}\) Any given instance \(I := (V, C)\) with parameter \(k(I) := k_0\) is mapped to an instance \(J\) which is a Turing machine \(M\) with parameter \(f(k_0)\), \(f\) to be fixed later.

Let \(B, G \subseteq V\). Define \(\Delta_{B, G}\) as the set of integers \(i \in [2, |C|]\) such that for the constraint \(C(i) := \text{CW}^b_{d,m} \langle \omega \rangle\), it holds that \(\omega([d]) = B\) and \(G \subseteq \omega([d + 1, d + m])\). Define \(\Lambda_{B, G}\) as
\[
\Lambda_{B, G} := \max_{C(i) := \text{CW}^b_{d,m} \langle \omega \rangle \in \Delta_{B, G}} |\omega^{-1}(G) \cap [d + 1, d + m]|.
\]

**Fact 5.2.** Let \(B, A \subseteq V\). Then \(|\Delta_{B, A}| = |\bigcup_{i \in A} \Delta_{B, \{i\}}|\) if and only if for every constraint \(\text{CW}^b_{d,m} \langle \omega \rangle\) in \(C\) such that \(\omega([d]) = B\), it holds that \(A \cap \omega([d + 1, d + m]) \neq \emptyset\).

To take advantage of the above fact, \(|\bigcup_{i \in A} \Delta_{B, \{i\}}|\) should be computed. The following claim shows how to do it efficiently.
Claim 5.3. For every \( B, A \subseteq V \), for any constraint \( CW_{d,m}^{[b]}(\omega) \) in \( C \) such that \( \omega([d]) = B \) implies \( |\omega([d+1, d+m]) \cap A| \leq b \), it holds that

\[
\left| \bigcup_{v \in A} \Delta_{B,\{v\}} \right| = \sum_{\substack{G \subseteq A \ \\ 0 < |G| \leq b}} (-1)^{|G|-1} |\Delta_{B,G}|.
\]

(18)

Proof.

\[
\left| \bigcup_{v \in A} \Delta_{B,\{v\}} \right| = \sum_{\substack{G \subseteq A \ \\ 0 < |G| \leq b}} (-1)^{|G|-1} \bigcap_{v \in G} \Delta_{B,\{v\}} \quad \text{ (inclusion-exclusion principle)}
\]

(19)

The last equality holds, because by assumption \( \forall G \subset V \ |G| > b \Rightarrow \Delta_{B,G} = \emptyset \).

The machine \( M \) of instance \( J \) can lookup the values \( |\Delta_{B,G}| \) and \( \Delta_{B,G} \) for \( |B| \leq k_0 \) and \( |G| \leq b + 1 \). \( M \) starts with a blank tape and operates in 3 steps and accepts if no rejection occurs.

Step 1. \( M \) chooses nondeterministically the set of variables \( A \subseteq V \) of size \( |A| \leq k_0 \).

Step 2. \( M \) iterates over all \( B, G \subseteq A \), such that \( |G| \leq b + 1 \), and rejects if \( \Delta_{B,G} > b \).

Step 3. \( M \) iterates over all \( B \subseteq A \), and performs the summation on the right side of Equation (18) and rejects if the result is not equal to \( |\Delta_{B,G}| \).

We show now that if instance \( I \) has a satisfying assignment \( E \), then the machine \( M \) accepts. Assignment \( E \) satisfies every constraint in \( C \), and so has size \( |E| = k_0 \). Consider the computation branch of \( M \), in which in Step 1 the set \( A \) is chosen equal to \( E \). In this branch, \( M \) rejects neither in Step 2, nor in Step 3 (by Fact 5.2 and Claim 5.3), therefore \( M \) accepts.

For the other direction, we prove that if machine \( M \) accepts, then in every accepting branch of computation, \( A \) is a satisfying assignment of \( I \). We have to only consider those constraints \( CW_{d,m}^{[b]}(\omega) \) in \( C \) such that \( \omega([d]) \subseteq A \). As there is no rejection in Step 2, we have \( |\omega^{-1}(A) \cap [d+1,d+m]| \leq b \). So the premises of Claim 5.3 hold for sets \( A \) and \( \omega([d]) \), and thus Equation (18) holds. Therefore, by Fact 5.2 as there is no rejection in Step 3, we have \( |\omega([d+1,d+m]) \cap A| \geq 1 \). So \( A \) satisfies the constraint.

More Details

For each variable \( v \in V \), there is a symbol \( \sigma_v \) in the alphabet of \( M \), with an alphabetical order over these symbols. Sets of variables are presented by the alphabetically sorted string of their symbols.

To do the summation in Equation (18) in Step 3, each time the calculated partial sum is added to the next summand. There are \( \binom{n}{b} \) summands, and each summand is an integer between \(-n \) and \( n \). Therefore the partial sum is always between \(-n \binom{k_0}{b+1} \) and \( n \binom{k_0}{b+1} \). Machine \( M \) has a symbol \( s_i \) for each integer \( i \) in this range and knows how to add two such integers.

By definition, \( |\Delta_{B,G}| < |G| \) and \( \Delta_{B,G} \leq n \). These values are stored in \( M \), only if \( |G| \leq \min(b+1, k_0) \) and \( |B| \leq k_0 \), and there is a constraint \( CW_{d,m}^{[b]}(\omega) \) in \( C \), such that \( \omega([d]) = B \) and \( G \subseteq \omega([d+1,d+m]) \).

Therefore the number of stored values is \( \leq n \sum_{i=0}^{b+1} \binom{n}{i} \). The lookups for other values will safely be answered by value 0. These values are stored and accessed by using the trie data structure, with the pairs \( (B, G) \) as the key, and the proper \( s_i \) symbol as the value. Notice that the length of the key is bounded by a fixed function of \( k_0 \). The above two arguments show that the runtime of the reduction is bounded by \( O(\sum_{i=0}^{b+1} \binom{n}{i})n^{O(1)} \).

In steps 2 and 3, \( M \) iterates over subsets of \( A \), all of size \( \leq k_0 \). As discussed above, the time needed for each lookup and summation is bounded by a function of \( k_0 \), therefore the runtime of \( M \) is bounded by \( f(k_0) \) for a fixed function \( f \).
Lemma 5.4. Let \( b \geq 0 \). Then \( \text{CSP}(W)_{k,t}(\text{CW}^{[b]})_k \) is in \( W[1] \).

Proof. We reduce the problem to Short NonDeterministic Turing Machine Acceptance. Let \( I := (V, C) \) be the given instance. Let \( I_1 := (V, C_1) \) and \( I_2 := (V, C_2) \) be the corresponding instances of \( \text{CSP}(W)_{k,t} \) and \( \text{CSP}(\text{CW}^{[b]})_k \), such that \( C = C_1 \cup C_2 \) and the parameter values of \( I_1 \) and \( I_2 \) are that of \( I \). Notice that \( V \) is the set of variables of \( I_1, I_2 \) and \( I \).

By Theorem 4.1, there is a nondeterministic Turing machine \( M_1 \) such that a set \( A_1 \subseteq V \) is a satisfying assignment of \( I_1 \) if and only if a branch of \( M_1 \) selects the set \( A_1 \) in Step 1 and accepts.

By Theorem 5.1, there is a nondeterministic Turing machine \( M_2 \) such that a set \( A_2 \subseteq V \) is a satisfying assignment of \( I_2 \) if and only if a branch of \( M_2 \) selects the set \( A_2 \) in Step 1 and accepts.

Consider the the nondeterminist Turing machine \( M \) described in the following. \( M \) nondeterministically simulates \( M_1 \) and then nondeterministically simulates \( M_2 \). A branch of \( M \) accepts if and only if \( M_1 \) and \( M_2 \) accept on the corresponding branches and \( A_1 = A_2 \).

If \( M \) accepts, then clearly \( A_1(= A_2) \) is a satisfying assignment of \( I \). On the other hand, if a set \( A \subseteq V \) is a satisfying assignment of \( I \), then \( A \) is a satisfying assignment of \( I_1 \) and \( I_2 \) by definition, therefor \( M \) has an accepting branch of computation where \( A_1 = A_2 = A \).

\[ \square \]

6 Partial sets and their Completions

Definition 6.1. Given a relation \( R \) (of arity say \( q \)), and a set \( T \subseteq \{q\} \), \( T \not\in R \), a completion of \( T \) is a minimal set \( U \) such that

\[
\begin{align*}
U & \in R \\
T & \subset U.
\end{align*}
\]

We denote the set of all completions of \( T \) by \( \text{compl}_R(T) \).

We say that a set \( T_1 \subseteq \{q\} \) is partial (to \( R \)) if \( T_1 \not\in R \) and any partial \( T_2 \subset T_1 \) has a completion that is a subset of \( T_2 \). We denote the set of all sets partial to \( R \) by \( \text{partial}(R) \).

Notice that if \( \emptyset \not\in R \), then \( \emptyset \in \text{partial}(R) \).

Fact 6.2. If \( T \) is a minimal set such that \( T \subseteq \{q\} \) and \( T \not\in R \), then \( T \) is partial.

Fact 6.3. Let \( W \in R \). If \( T \subset W \) and \( T \not\in R \), then there is a set \( U \subseteq W \) such that \( U \in \text{compl}_R(T) \).

It is not hard to see that for a set \( D \subseteq \{q\} \), we have \( D \in R \) if and only if for every \( T \not\in R \) such that \( T \subseteq D \), there is \( U \in \text{compl}_R(T) \) such that \( U \subseteq D \). But we can be much more efficient: \( D \in R \) if and only if for every \( T \in \text{partial}(R) \) such that \( T \subseteq D \), there is \( U \in \text{compl}_R(T) \) such that \( U \subseteq D \). The reduction in the proof of the following theorem is based on this idea.

Theorem 6.4. Let \( \Gamma = \{R_i\}_{i \in \mathbb{N}} \) be a (possibly infinite) constraint language and there be an integer \( d \geq 1 \) such that for any \( R \in \Gamma \), for every \( T \in R \), it holds \( |T| \leq d \). Then \( \text{CSP}(\Gamma)_k \) is in \( W[1] \).

Proof. We show that \( \text{CSP}(\Gamma)_k \) is fpt-reducible to \( \text{CSP}(W)_{t,k \leq (\text{CW}[2^d])}_{k \leq} \). The result follows by Lemma 5.4 and Lemma 2.1. Given an instance \( I := (V_1, C) \) with parameter \( k(I) = k_0 \), we construct an instance \( J := (V_1 \cup V_2, C') \) as follows.

First notice that because \( \Gamma \) is fpt membership checkable, for every \( i \in \mathbb{N} \), the set \( \text{partial}(R_i) \) can be computed in time \( O(r^d)(\log i)^{O(1)} \), where \( r := \text{arity}(R_i) \). Also for each \( T \in \text{partial}(R) \) the set \( \text{compl}_R(T) \) can be computed in time \( O(r^d)(\log i)^{O(1)} \).

For each constraint \( C(i) := R(\omega_i), i \in [2, |C|] \), and for each \( T \in \text{partial}(R) \), introduce the new variable \( \lambda_{\omega_i}(T) \) and the set of new variables \( B \),

\[ B := \{\lambda_{\omega_i}(U) | U \in \text{compl}_R(T)\} \tag{21} \]
(if they are not already introduced), and add a constraint $CW^{[2^2]}_{1,|B|}([\omega_2])$ to $C'$, where $\omega_2(1) = \lambda_{\omega_1(T)}$ and $\omega_2([|B|+1]) = B$.

For each new variable $\lambda_E$ introduced above ($E \subseteq V_1$, a set of variables), add a constraint $CW^{[1]}_{|E|,1}([\omega])$ to $C'$ where $\omega([|E|]) = E$ and $\omega([|E|+1]) = \lambda_E$. Also, for every variable $x \in E$, add the constraint $CW^{[1]}_{1,1}([\omega])$ to $C'$, where $\omega(1) = \lambda_E$ and $\omega(2) = x$. We call these binding constraints (notice that for every $b \geq 0$, it holds $CW^{[1]}_{b,1} \in CW^{[2^2]}$).

Set $V_2$ to be the set of all newly introduced variables. Add to $C'$ the constraint $W^{[k_0]}_{|V_1|}([\omega_3])$ where $\omega_3([|V_1|]) := V_1$. Finally, add the constraint $C'(1) := W^{[k_0+2^k_0]}_{|V_1|+|V_2|}([\omega_4])$, where $\omega_4([|V_1|+|V_2|]) := V_1 \cup V_2$.

This sets parameter $k_0$ of $J$ to $k_0 + 2^k_0$.

**Fact 6.5.** For two sets $D \subseteq V_1$ and $Q \subseteq V_2$, let $D \cup Q$ satisfy $J$. Then the binding constraints in $C'$ ensure that

$$\forall \lambda_E \in V_2 \quad \lambda_E \in Q \iff E \subseteq D.$$  \hfill (22)

By Fact 6.5, it is enough to prove the following claim to show that $I$ has a satisfying assignment if and only if $J$ has a satisfying assignment.

**Claim 6.6.** Let sets $D \subseteq V_1$ and $Q \subseteq V_2$ be such that $Q = \{\lambda_E | \lambda_E \in V_2, E \subseteq D\}$. Then the assignment $D$ satisfies $I$ iff $D \cup Q$ satisfies $J$.

**Proof.** If $D$ is a solution of $I$, let $\psi$ be a constraint in $C'$. If $\psi$ is a binding constraint, then it is trivially satisfied by $D \cup Q$. Otherwise, let $\psi$ be a constraint $CW^{[2^2]}_{1,|B|}([\omega])$ which corresponds to some constraint $\omega_1 \in C$ and a set $T \in \text{partial}(R)$ (with set $B$ as defined in the construction above). So $\omega_2(1) = \lambda_{\omega_1(T)}$ and $\omega_2([|B|+1]) = B$. Now if $\omega_1(T) \not\subseteq D$, then $\lambda_{\omega_1(T)} \not\in Q$ and assignment $D$ trivially satisfies $\psi$. Else if $\omega_1(T) \subseteq D$, first, $\lambda_{\omega_1(T)} \in Q$. Secondly, we apply the Fact 6.3. We have $\omega_1^{-1}(D) \in R$, $\omega_1^{-1}(D) \subseteq D$ and $\omega_1^{-1}(D)$, therefore there is a $U \in \text{compl}_R(T)$ such that $U \subseteq \omega_1^{-1}(D)$, which means $\omega_1(U) \subseteq D$ (notice that the number of such $U$ is at most $2^2$). Therefore variable $\lambda_{\omega_1(U)}$ in set $V_2$ exists and $\lambda_{\omega_1(U)} \in Q$. It follows that $\psi$ is satisfied by assignment $D \cup Q$.

For the other direction, assume for the sake of contradiction that $D \cup Q$ satisfies $J$, but $D$ does not satisfy $I$. The first constraint $C(1)$ is trivially satisfied, so let $R(\omega)$ be another constraint of $C$ such that $\omega^{-1}(D) \not\in R$. By Definition 6.1, there is $T \in \text{partial}(R)$ such that $T \subseteq \omega^{-1}(D)$ and for every $U \in \text{compl}_R(T)$, it holds that $U \not\subseteq \omega^{-1}(D)$. There is, by construction, variable $\lambda_{\omega(T)} \in V_2$, and $\lambda_{\omega(T)} \in Q$. And either $\text{compl}_R(T) = \emptyset$ or for every $U \in \text{compl}_R(T)$, variable $\lambda_{\omega(U)} \not\in Q$. Therefore, the constraint $\psi$ in $J$ that (by construction) corresponds to $R(\omega)$ and $T$, is not satisfied by assignment $D \cup Q$, a contradiction.

**Corollary 6.7.** Let $d \geq 0$. Then CSP($W[d]$)$_k \in W[1]$.

**References**

[1] V. Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Solving linear equations parameterized by Hamming weight. In Proceedings of 9th International Symposium on Parameterized and Exact Computation (IPEC), number 8894 in LNCS, pages 39–50. Springer, 2014. doi:10.1007/978-3-319-13524-3_4

[2] Andrei A. Bulatov and Dániel Marx. Constraint satisfaction parameterized by solution size. sicomp, 43(2):573–616, 2014. doi:10.1137/120882160
[3] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. On the parameterized complexity of short computation and factorization. *Arch. Math. Log.*, 36(4-5):321–337, 1997. URL: [http://dx.doi.org/10.1007/s001530050069](http://dx.doi.org/10.1007/s001530050069), doi:10.1007/s001530050069.

[4] Marco Cesati. Perfect Code is W[1]-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002. URL: [http://dx.doi.org/10.1016/S0020-0190(01)00207-1](http://dx.doi.org/10.1016/S0020-0190(01)00207-1).

[5] Marco Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[6] Marek Cygan, Fedor Fomin, Bart MP Jansen, Łukasz Kowalik, Danilo Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Open problems for fpt school 2014. URL: [http://ftpschool.mimuw.edu.pl/opl.pdf](http://ftpschool.mimuw.edu.pl/opl.pdf), 2014.

[7] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995. doi:10.1016/0304-3975(94)00097-3.

[8] Rod G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer, 1999.

[9] Rod G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999. doi:10.1137/S009753979732571.

[10] Rodney G Downey, Michael R Fellows, and K Regan. Descriptive complexity and the W-hierarchy. *Proof Complexity and Feasible Arithmetic*, 39:119–134, 1998.

[11] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, 2006.

[12] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48(6):1184–1206, 2001. doi:10.1145/504794.504798.

[13] Michael T. Hallett and Harold T. Wareham. The parameterized complexity of some problems in logic and linguistics. In Anil Nerode and Yuri Matiyasevich, editors, *Logical Foundations of Computer Science, Third International Symposium, LFCS’94, St. Petersburg, Russia, July 11-14, 1994, Proceedings*, volume 813 of *Lecture Notes in Computer Science*, pages 89–100. Springer, 1994. URL: [http://dx.doi.org/10.1007/BFb0554107](http://dx.doi.org/10.1007/BFb0554107), doi:10.1007/3-540-58140-5_10.

[14] Dániel Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005. doi:10.1007/s00037-005-0195-9.

[15] Christos Harilaos Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Mass., 8 1995.