On the Computational Limits of Infinite Satisfaction

Stefan Dantchev*
Dept. of Computer Science, Univ. of Durham
s.s.dantchev@durham.ac.uk

Frank D. Valencia†
CNRS, LIX École Polytechnique de Paris
frank.valencia@lix.polytechnique.fr

ABSTRACT

We study the *computational limits* of Constraint Satisfaction Problems (CSP's) allowing *infinitely*, or *unboundedly*, many *indexed* variables as in, e.g., $x_i > x_{i+2}$ for each $i = 1, 2, \ldots$. We refer to these CSP's as *Infinite CSP's (ICSP's)*. These problems arise in contexts in which the number of variables is unknown a priori as well as in optimization problems wrt the number of variables satisfying a given finite set of constraints.

In particular, we investigate the decidability of the satisfiability problem for ICSP's wrt (a) the first-order theory specifying the *indices* of variables and (b) the dimension of the indices. We first show that (1) if the indices are one-dimensional and specified in the theory of the natural numbers with linear order (the theory of $(\mathbb{N}, 0, succ, <)$) then the satisfiability problem is *decidable*. We then prove that, in contrast to (1), (2) if we move to the two-dimensional case then the satisfiability problem is *undecidable* for indices specified in $(\mathbb{N}, 0, succ, <)$ and even in $(\mathbb{N}, 0, succ)$. Finally, we show that, in contrast to (1) and (2), already in the one-dimensional case (3) if we also allow addition, we get undecidability. I.e., if the one-dimensional indices are specified in *Presburger arithmetic* (i.e., the theory of $(\mathbb{N}, 0, succ, <, +)$) then satisfiability is *undecidable*.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Logic and constraint programming*; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems—*Decision problems*

General Terms

Theory

*The contribution of S. Dantchev has been partly supported by the Nuffield Foundation award to newly appointed lecturers in Science, Engineering and Mathematics NAL/00775/G.

†The contribution of F. Valencia has been partly supported by the research team **Comète** from INRIA Futures.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

Keywords

Constraint Satisfaction Problems, Decidability, Open CSP, Infinite CSP

1. INTRODUCTION

In many constraint applications the number of variables may not be not known a priori, hence it can naturally be modeled as being *infinite (or unbounded)* as done, e.g., in [3 4 8]. This particularly applies to today's applications due to the advent of the Internet. For instance, we may want to specify constraints over the resources of a high-scale distributed network and these resources may be continuously added to the network. In fact, the study of algorithms for this kind of CSP's and their applications have been considered in several work, e.g., [3 4 7].

Nevertheless, to our knowledge, there is no work on the computational limits for the satisfiability of CSP's allowing infinitely (or unboundedly) many variables. I.e., studies establishing, for interesting classes of these problems, whether their satisfiability can actually be *computationally* determined. In traditional CSP's this is not a issue since everything in them has a finite nature. But it is certainly an issue when you allow an infinite (or unbounded) number of variables, even when taking on values from finite domains.

For example, a designer may formulate CSP's for unbounded number of variables by using constraint specifications over an infinite number of variables of the form, e.g., $x_i > x_{i+1}$ for $i = 2, 4, 6, \ldots$ or with two-dimensional indexes like in $x_{i,j} = x_{i,j-1}$ for all $i, j > 0$. This way, the designer wishes to express that his/her specification must be satisfiable for *any number* of variables. It may simply turn out, however, that determining the satisfiability of a given specification of this kind is computationally impossible, i.e. *undecidable*.

Another example of the relevance of *decidability* for CSP's with unbounded number of variables has to do with *optimization problem* of the kind suggested by [2]. For example, suppose you are asked to provide a *terminating* algorithm $\mathcal{A}$ such that: Given a CSP P with an unbounded number of variables $x_0, x_1, \ldots$, $\mathcal{A}$ should return the maximal number $m \in \mathbb{N}$ of variables $x_0, x_1, \ldots, x_m$ such that P is satisfiable. Additionally, if such m does not exist, $\mathcal{A}$ must report this. One might suggest an algorithm $\mathcal{A}$ that tries incrementally different values of m until it finds the first one after which the problem is no longer satisfiable. Notice, however that some given P may be satisfiable for *any number* number of variables—i.e., the maximal m does not exist. Hence, the suggested algorithm may actually never terminate on such a P. In fact, for some natural classes of ICSP's the algorithm $\mathcal{A}$ *cannot* exist because, as we shall show in this paper, decidability of their satisfiability is undecidable thus contradicting the existence of $\mathcal{A}$.

Approach. Constraint problems with unbounded number of variables are typically specified with expressions generating constraints over *infinitely many indexed* variables; e.g., $x_i > x_{i+1}$ for $i = 0, 2, \ldots$. In this paper we shall model CSP's that allow constraints on *infinitely many indexed variables* but each taking on values from a *finite* domain. We call these problems *Infinite CSP's* (ICSP's). In particular we investigate the decidability of the *satisfiability* (i.e., the existence of solutions) for ICSP's wrt two properties on the indices of the variables. Namely, the *index dimension*; e.g., of the form x_i (one dimensional), or $x_{i,j}$ (two dimensional), and the *language to specify expressions over the indices*; e.g., using index comparison and summations as in "$c(x_i, x_{i+j})$ for all $j < i$".

Results. The results we present in this paper are the following:

1. If the ICSP's variable indices are one-dimensional and specified in the theory of the natural numbers with linear order (the theory of $(\mathbb{N}, 0, succ, <)$) then the ICSP satisfiability is *decidable*. An instance of this class of problems is an ICSP with constraints of the form $x_i < x_{i+1}$ for $i = 0, 2, 4, \ldots$.

2. In contrast to (1) we prove that for the two-dimensional case, ICSP satisfiability is *undecidable* for indices specified in $(\mathbb{N}, 0, succ)$ and hence also in $(\mathbb{N}, 0, succ, <)$—E.g., ICSP's with constraints of the form $x_{i,j} = x_{i-1,j+1}$ for each $i, j = 0, 1, 2, \ldots$.

3. If the indices are specified in *Presburger arithmetic* (i.e., the theory of $(\mathbb{N}, 0, succ, <, +)$ then satisfiability is *undecidable* already in the one-dimensional case. An instance of this class of ICSP's may have constraints such as $x_i < x_{2i}$ for $i = 1, 2, \ldots$—where $2i \stackrel{\text{def}}{=} i + i$.

Another noteworthy aspect of our work is that result (1) is obtained by establishing a connection between ICSP's and Finite-State Automata. Several theoretical and implementational work, specially in the area of Verification have been carried out for Finite-State Automata. Therefore, we may hope to benefit from this connection both for theoretical and application purposes in the study of ICSP. For example, we may be able to tailor the successful MONA verification tool [11] to deal with some classes of ICSP's.

Preliminaries

We assume that the reader is familiar with basic concepts of Computability and Logic such as Turing Machines, Finite-State Automata, Decidability, countable sets, Truth and first-order formulae. A *structure* $\mathcal{A} = (|\mathcal{A}|, \Phi)$ consists of a *universe* $|\mathcal{A}|$ and an *interpretation* function Φ whose domain is a set of constant, function, and predicate symbols $sig(A)$; the *signature* of $\mathcal{A}$. The function Φ interprets each constant, function and predicate symbol in $sig(\mathcal{A})$, according to its *rank* (also called *arity*), as an element, function, or relation over $|\mathcal{A}|$ in the obvious way. Typically, the function Φ is omitted and replaced by its domain whenever Φ is clear from the context. For example, the structure of arithmetic is $(\mathbb{N}, 0, 1, +, <, \times)$ where the constant symbols $0, 1$ are respectively interpreted as the numbers zero and one, the binary function symbols $+, \times$ as natural number addition and multiplication, and $<$ is interpreted as the less relation on the natural numbers. An n-sorted structure is a structure with n universes (or sorts). In such structures the symbols and variables are assigned sorts and interpreted accordingly.

Given a structure $\mathcal{A}$, the *first-order theory* of $\mathcal{A}$, $Th(\mathcal{A})$, is the set of all first-order formulae over $sig(\mathcal{A})$ that are true in $\mathcal{A}$. The set $Th(\mathcal{A})$ is *decidable* iff there exists an effective procedure that will determine whether $\phi \in Th(\mathcal{A})$ for any first-order formula ϕ over $sig(\mathcal{A})$. A structure is *countable* iff its universe is countable.

Related Work

CSP's with unboundedly many variables, their relevance, and algorithms for them have been reported in other works, e.g., [4 7]. To our knowledge, however, there is no work on the computational limits of determining the satisfiability of these problems; i.e., whether it is computational possible.

An infinite CSP, and more precisely an infinite SAT, was studied by Freedman [8]. He used concrete groups as indexing structures and tried to establish a connection between easy (hard) problems in the finite case and decidable (undecidable) problems in the infinite case. More recently, in [3] the computational complexity of infinite CSP's was studied in the algebraic settings. In both papers the emphasis is on the *complexity of the ICSP*, while in the present paper we are mainly concerned with *decidability issues*.

There has been some work on dynamic variable generation, e.g., [4]. Loosely speaking, this approach can be used to solve an infinite CSP by "approximating" it by a sequence of finite CSP's. Note, though, that there are satisfiable CSP's whose finite fragments are unsatisfiable. Toy examples include the Pigeon-Hole Principle (there is a bijection between a set and the same set with one element removed) and the Least Number Principle (there is a partial order that has no minimal element). Thus the dynamic variable generation would fail on these examples.

The authors in [10] showed some evidence that, in general, it would be hard to find a connection between the computational complexity of a finite CSP and the recursive complexity of its infinite counter-part. Thus we entirely concentrate on the decidability barrier that depends, in our case, on the indexing structure as well as the arity of indices, and completely ignore the complexity issues.

2. ICSP'S

Here we define our framework for *modeling* CSP's allowing infinitely (unboundedly) many indexed variables; ICSP's. We find it convenient to parameterize ICSP's with an *indexing structure* that specifies the indices of their variables.

An infinite constraint satisfaction problem (ICSP) consists of *finitely* many symbols $x, y, \ldots, \in X$ which can be indexed to generate *infinitely many indexed variables*. We assume that each $x \in X$ is *ranked*; i.e., it has a *rank* (or *arity*), denoted as $rank(x)$, that indicates the number of indexes the symbol can take, e.g., if $rank(x) = 2$ then we can index x as $x_{1,3}$—to avoid too many subindex levels we should sometimes write e.g. $x(1, 3)$ to mean $x_{1,3}$. In an ICSP, the indexed variables take values from a given finite domain Σ and must satisfy the constraints specified in a given set R, e.g., $x_i > x_j$ for all $j < i$.

Now, in order to provide a finite representation of ICSP's we shall use *first-order theories* as the language to specify the indices of variables—clearly, such theories must be decidable to hope for decidable families (wrt to satisfiability) of ICSP's. Typically, the indices are natural numbers for one-dimensional indices, or tuples of natural numbers for the multi-dimensional case. Thus, it seems natural to use *countable structures* with decidable theories. Examples of such structures include among others:

- $(\mathbb{N}, 0, succ, <, +)$ (*Presburger arithmetic*),
- $(\mathbb{N}, 0, succ, <)$ (*Linear-order arithmetic*),
- $(\mathbb{N}, 0, succ)$ (*Successor arithmetic*),
- $(\{0, \ldots, n-1\}, 0, succ, <, +, \times)$ (*Arithmetic modulo n*).

All in all, ICSP's are tuples of the form $P = \langle \mathcal{I}, X, \Sigma, R \rangle$. Intuitively, P specifies a problem with an infinite set of *indexed variables* $Vars(P)$ generated by indexing the symbols in X with (sequences of) elements of $|\mathcal{I}|$ where $\mathcal{I}$ is a countable structure with decidable $Th(\mathcal{I})$. The indexed variables in $Vars(P)$ take on values from the finite *domain* Σ while complying with the *constraints* given by the *constraint-patterns* in R. Now, a constraint-pattern specifies what indexed variables its constraint c should be applied to. For example, we shall use constraints patterns such as

$$\langle \forall_i \triangleright i > 42, \exists_j \triangleright j < i \; ; \; c[x(i), y(j)] \rangle$$

to mean:

"For each $i > 42$, there exists $j < i$ such that $c(x_i, y_j)$ holds."

The expression $\forall_i \triangleright i > 42$, an *universal index-generator*, produces all indices i greater than forty-two, and $\exists_j \triangleright j < i$, an *existential index-generator*, produces some index j such that $i < j$.

The above intuition about ICSP's is made precise by the following definition.

DEFINITION 2.1 (ICSP'S). *An ICSP $P = \langle \mathcal{I}, X, \Sigma, R \rangle$ is a tuple satisfying the following:*

- *$\mathcal{I}$ is a countable structure and $Th(\mathcal{I})$ is decidable; the* indexing structure.
- *X is a finite set of ranked symbols; the* indexable-variables.
- *Σ is a finite set of constants; the* domain of values.
- *R is a finite set of* constraint patterns. *A constraint pattern has the form $\langle G[i_1], \ldots, G[i_n]; c[x_1(s_1), ..., x_n(s_n)] \rangle$ where $c \subseteq \Sigma^n$ is an n-ary relation called* constraint, *and for $k = 1, \ldots, n$:*
  - *$G[i_k] = \mathbf{Q}_{i_k} \triangleright \phi_k$ with $\mathbf{Q} \in \{\forall, \exists\}$ and $\phi_k(i_1, \ldots, i_k)$ a formula over $sig(\mathcal{I})$ with free-variables $i_1, \ldots, i_k$.*
  - *$x_k \in X$ and $s_k \in \{i_1, ..., i_n\}^{rank(x_k)}$.*

  Each $G[i_k]$ is called an index-generator *of i_k;* universal *if it takes the form $\forall_{i_k} \triangleright \phi$,* existential *otherwise.*

Define $Vars(P)$, the (indexed) variables *of P, as the set of all $x(t)$, also written as x_t, with $x \in X$ and $t \in |\mathcal{I}|^{rank(x)}$.* □

We now describe some conventions on constraint expressions to simplify the presentation of ICSP's.

CONVENTION 2.2. *For any constraint c, we assume a predicate $c(x_1, \ldots, x_n) = \top$ iff $(x_1, \ldots, x_n) \in c$. We shall take the liberty of expressing constraints as first-order formulae over the indexed variables of the underlying ICSP. Also, if x_t is an indexed* boolean variable *(i.e., the domain of the underlying ICSP is boolean), we find it convenient to write x_t and $\neg x_t$ to denote the constraints $x_t = \top$ and $x_t = \bot$, respectively. Furthermore, we shall often assume implicit the meaning of symbols and expression when clear from the context. E.g. if the underlying domain is $\Sigma = \{0, 1, 2\}$, we implicitly assume $<$ denotes $\{(0, 1), (0, 2), (1, 2)\}$.*

Let us now illustrate a very simple ICSP example.

EXAMPLE 2.3. *Consider the constraint problem with variables $x_1, x_2, \ldots$ taking on values in $\{0, 1\}$ and satisfying $x_i > x_{i+1}$ for each even i. This problem can be specified as $P = \langle \mathcal{I}, X, \Sigma, R \rangle$, where $\mathcal{I} = (\mathbb{N}, 0, succ, <, +)$, $X = \{x\}$ with the rank of x being 1, $\Sigma = \{0, 1\}$, and $R = \{\langle \forall i \triangleright even(i), \forall j \triangleright \phi; > (x(i), x(j)) \rangle\}$ with $even(i) \stackrel{\mathsf{def}}{=} \exists_l (l + l = i)$ and $\phi(i, j) \stackrel{\mathsf{def}}{=} j = i + 1$.* □

A solution of an ICSP $P = \langle \mathcal{I}, X, \Sigma, R \rangle$ is an assignment of values from the domain Σ to the variables $Vars(P)$ that satisfies the constraints specified by the constraints-patterns in R. Formally,

DEFINITION 2.4 (SATISFIABILITY). *Let $P = \langle \mathcal{I}, X, \Sigma, R \rangle$ be an ICSP. Let $\mathcal{I}_P$ be the (two-sorted) structure that results from extending $\mathcal{I}$ with a universe Σ, the symbols in X and $Vars(P)$, and, for each constraint c in R, a symbol $\bar{c}$, interpreted as the relation c.*

An interpretation *(or* assignment*) $v : Vars(P) \to \Sigma$ assigns to each $x(t) \in Vars(P)$ a value in Σ. An assignment v is said to be* solution of *the ICSP P if and only if for every constraint pattern $\langle G[i_1], \ldots, G[i_n]; c[x_1(s_1), ..., x_n(s_n)] \rangle \in R$, v satisfies, in the structure $\mathcal{I}_P$, the formula*

$$\mathbf{Q}_1 i_1 (\phi_1 \bullet_1 \mathbf{Q}_2 i_2 (\phi_2 \bullet_2 \ldots \mathbf{Q}_n i_n (\phi_n \bullet_n \bar{c}(x_1(s_1), \ldots, x_n(s_n)))))$$

where $\bullet_k = \Rightarrow$ if $\mathbf{Q}_k = \forall$ else $\bullet_k = \wedge$.

An ICSP is satisfiable *if and only if it has a solution.* □

EXAMPLE 2.5. *The ICSP P in Example 2.3 is satisfiable. Verify that the assignment $v(x_0) = 1, v(x_1) = 0, v(x_2) = 1, v(x_3) = 0, v(x_4) = 1, \ldots$ is a solution of P.*

3. SOME SIMPLE ICSP'S EXAMPLES

Interesting applications of CSP with unbounded number of variables can be found [3 4 7]. Here, in order make the reader familiar with our ICSP representation, we we shall show how to express some musical and a infinite version of well-known recreational mathematics as ICSP's.

In what follows, we simplify further the notation: every generator of the form $\mathtt{Q}_x \triangleright \top$, i.e. there are no restrictions on the index x, is denoted simply by $\mathtt{Q}_x$.

Music Sequences. The authors in [13] argued for ICSP's in the context of computer music via constraint specifications. Constraints are defined on the arbitrary long sequences of chords that can be generated. Each chord represents the notes to be played at the same time. The highest voice is the sequence that results from taking the highest notes in the sequence of chords. The lower voice is defined analogously. A standard constraint is that the highest voice goes up at time i iff the lowest voice does not do it at the same time. This standard constraint can be defined as an ICSP as follows $\langle \forall i; h_i < h_{i+1} \Leftrightarrow \neg(l_i < l_{i+1}) \rangle$. The meaning of h_i and l_i is obvious; the highest and lowest note, respectively, played at time i. The underlying indexing structure of this problem is Linear-order Arithmetic $(\mathbb{N}, 0, succ, <)$.

Infinite N-Queens. The second problem we consider is *the infinite n-queens problem*. We have an infinite chess-board (rectangular grid in the plane) and want to place an infinite number of queens so that: there is exactly one queen in each row and each column; there are no two queens in any diagonal. The infinite n-queens problem can be represented by the following ICSP with indexed boolean variables of the form q_{ij} and the following constraint patterns :

$$\begin{aligned}
\langle \forall_i \exists_j \quad ; \quad & q_{ij} \rangle \\
\langle \forall_{i,u,v} \triangleright (u \neq v) \quad ; \quad & q_{iu} \Rightarrow \neg q_{iv} \rangle \\
\langle \forall_i \exists_j \quad ; \quad & q_{ji} \rangle \\
\langle \forall_{u,v,j} \triangleright (u \neq v) \quad ; \quad & q_{uj} \Rightarrow \neg q_{vj} \rangle \\
\langle \forall_{u,v,i,j} \triangleright (u + v = i + j) \wedge (u \neq i) \quad ; \quad & q_{uv} \Rightarrow \neg q_{ij} \rangle \\
\langle \forall_{u,v,i,j} \triangleright (u - v = i - j) \wedge (u \neq i) \quad ; \quad & q_{uv} \Rightarrow \neg q_{ij} \rangle
\end{aligned}$$

The meaning of q_{ij} is obvious: there is a queen in the ith row and jth column iff $q_{ij} = \top$. The first line says there is at least one queen in each row, while the second one says that there are not two queens in the same row; the first two constraints altogether express the fact that there is exactly one queen in each row. The third and fourth lines express the same property for columns. The las two constraints say that there is no diagonal containing two queens. The underlying indexing structure is Presburger Arithmetic.

4. DECIDABILITY RESULTS

In this section we present (un)decidability results for the satisfiability of ICSP's.

CONVENTION 4.1. *We shall say that an ICSP $\langle \mathcal{I}, X, \Sigma, R\rangle$ is k-dimensional $k \geq 1$ iff each symbol in X has at most rank k and at least rank one. Also, we shall often say that a certain family of ICSP's are (un)decidable to mean that the satisfiability problem for such a family is (un)decidable.*

First, we state the decidability for the family of one-dimensional ICSP's $P = \langle \mathcal{I}, X, \Sigma, R\rangle$ with indices specified in the theory of the natural numbers with linear order; i.e., $\mathcal{I} = (\mathbb{N}, 0, succ, <)$. We then show that if we move to the two-dimensional case, ICSP's with linear-orders become undecidable. We conclude by showing that if we also allow summation(i.e. Presburger arithmetic $\mathcal{I} = (\mathbb{N}, 0, succ, <, +)$) then ICSP's are undecidable already in the one-dimensional case.

4.1 Linear-Order ICSP's: One Dimension

We shall state that one-dimensional ICSP's with linear-order indexing structures can be characterized by Büchi automata [5].

Büchi FSA. Recall that Büchi automata are ordinary finite-state automata (FSA) with an acceptance condition for infinite (or ω) sequences: an ω sequence is accepted iff the automaton can read it from left to right while visiting a final state infinitely often. The language recognized by a Büchi automaton B is denoted by $\mathcal{L}(B)$. Regular ω-languages are those recognized by Büchi FSA.

We shall use the following notation.

NOTATION 4.2. *Let $P = \langle \mathcal{I}, X, \Sigma, R\rangle$ be a one-dimensional ICSP. Suppose that $X = \{x_1, \ldots, x_n\}$ and that $v : Vars(P) \to \Sigma$ is an assignment for the indexed variables of P. We use $v(X_t)$ where $t \in |\mathcal{I}|$ to denote $(v(x_{1_t}), \ldots, v(x_{n_t}))$; i.e., the sequence of values assigned to variables that are indexed with t.*

The idea of the characterization of linear-order one-dimensional ICSP's is that every solution v of any such an ICSP can be viewed as an ω-sequence of the form $v(X_0).v(X_1)\ldots.$ exhibiting (ω) regularities. Let us illustrate this with the following example.

EXAMPLE 4.3. *Consider a problem with variables $x_0, x_1, \ldots$ whose domain is $\Sigma = \{0, 1\}$, and constraints $x_i \neq x_{i+1}$ for each $i = 0, 1, 2, \ldots$. Clearly, this problem can be expressed as a one-dimensional ICSP P with indexing structure $(\mathbb{N}, 0, succ, <)$ and set of indexable symbols $X = \{x\}$. Now, v is a solution of P iff $v(X_0).v(X_1)\ldots.$ is accepted by the Büchi automaton in Figure 1.* □

The following lemma states the Büchi automata representation of the ICSP's under consideration.

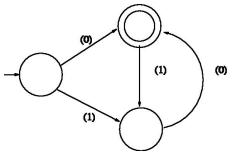

Figure 1: A Büchi characterization of the ICSP in Example 4.3.

LEMMA 4.4 (BÜCHI REPRESENTATION). *Suppose that $P = \langle (\mathbb{N}, 0, succ, <), X, \Sigma, R\rangle$ is a one-dimensional ICSP. One can effectively construct a Büchi FSA B_P such that $\mathcal{L}(B_P)$ is (isomorphic to)*

$$\{v(X_0).v(X_1)\ldots. \mid v \text{ is a solution of } P\}.$$

PROOF. See [6]. □

As a corollary of Lemma 4.4 and the decidability of the emptiness problem for Büchi automata [14], we get the decidability of satisfiability for the ICSP's under consideration.

THEOREM 4.5. *Let $P = \langle (\mathbb{N}, 0, succ, <), X, \Sigma, R\rangle$ be a one-dimensional ICSP. The question of whether P has a solution is decidable.*

4.2 Linear-Order ICSP's: Two-Dimensions

Here, we show that in contrast to one-dimensional ICSP's, the satisfiability problem for two-dimensional ones with linear-order indexing structure is undecidable. In fact, we prove something stronger: The problem is undecidable for two-dimensional ICSP's whose indexing structure is simply $(\mathbb{N}, 0, succ)$.

The result can be obtained via reduction from one of the classic undecidable problems in computability: the *Halting Problem for Turing-Machines*. The key idea of the reduction is to use a variable $x_{i,t}$ for each $i, t = 0, 1, \ldots$ to represent the state of the i-th cell in the tape of the machine at time t. Constraints can then be set up to express valid transitions between configurations.

The above-mentioned reduction suggests that linear-order two-dimensional ICSP's are very expressive as they are capable of specifying arbitrary Turing computations. This is to be contrasted with the expressiveness of linear order one-dimensional ICSP's; from Lemma 4.4 it does not go beyond of that of Büchi FSA.

Our reduction lemma can be stated as follows:

LEMMA 4.6 (TURING REDUCTION). *Let M be a Turing machine. One can effectively construct a two-dimensional ICSP P_M with indexing structure $(\mathbb{N}, 0, succ)$ such that:*

$$P_M \text{ has a solution iff } M \text{ does not halt}.$$

PROOF. See [6]. □

As an immediate consequence of the previous lemma and the undecidability of the Halting problem we get undecidability of two-dimensional ICSP's with $(\mathbb{N}, 0, succ)$. Hence, two-dimensional ICSP's with linear-order as indexing structure are also undecidable.

THEOREM 4.7. *Let $P = \langle \mathcal{I}, X, \Sigma, R\rangle$ be a two-dimensional ICSP with indexing structure $\mathcal{I} \in \{(\mathbb{N}, 0, succ), (\mathbb{N}, 0, succ, <)\}$. The question of whether P has a solution is undecidable.*

4.3 Presburger ICSP's: One Dimension

In this section we shall prove that Presburger one-dimensional ICSP's are undecidable wrt satisfiability, even if we only have *a single single-indexed variable, all constraint patterns are universal*

and *each constraint is a* 2*-clause*, i.e. a disjunction of at most two literals.

Now, it is well known that Presburger Arithmetic with a single (uninterpreted) unary predicate is undecidable (in fact Π_1^1 complete) [9]. It is also known that certain temporal logics based on Presburger Arithmetic are undecidable, too [1]. Unfortunately neither proof applies to our setting.

Two-counter Machines. Our proof is a reduction from the *Halting problem for two-counter machines*. Two-counter machines are a universal model of computation, i.e. equivalent to Turing machines. As in the case of two-dimensional linear-order ICSP's, the reduction also states that Presburger ICSP's are very expressive as implies they are capable of specifying arbitrary Turing computations. However, in Presburger this already happens for one-dimensional ICSP's.

Before stating the reduction, we need to recall how two-counter machines are defined. They were introduced by Minsky in [12] and are also knows as *Minsky machines*. There are two registers (called *counters*) c_1 and c_2; each of them holds a natural number. The "program" of the machine is a sequence of m instructions $\alpha_0, \alpha_1, \ldots \alpha_{m-1}$, each instruction being of the forms

1. Add 1 to c_i and go to α_j ($i \in \{1,2\}$ and $0 \leq j < m$).
2. if $c_i = 0$ go to α_j else subtract 1 from c_i and go to α_k ($i \in \{1,2\}$ and $0 \leq j,k < m$).

The machine always starts at α_0 with $c_1 = 2^x$ and $c_2 = 0$ where $x \in \mathbb{N}$ is the input. It stops only if it reaches α_{m-1} with $c_1 = 2^y$ for some $y \in \mathbb{N}$ and $c_2 = 0$; y is the result of the computation.

We can now state the reduction lemma:

LEMMA 4.8 (MINSKY REDUCTION). *Suppose that $\mathcal{M}$ is a Minsky machine with a sequence of m instructions $\alpha_0, \ldots \alpha_{m-1}$ and a natural number n. It is possible to effectively construct a one-dimensional ICSP $P_{\mathcal{M}}$ with $(\mathbb{N}, 0, succ, <, +)$ as indexing structure such that:*

$P_{\mathcal{M}}$ is satisfiable if and only if $\mathcal{M}$ does not halt on the input n.

PROOF. See [6]. □

We can now state the undecidability of Presburger one-dimensional ICSP's which follows immediately from Lemma 4.8 and the undecidability of the Halting problem for two-counter machines.

THEOREM 4.9. *Let $P = \langle(\mathbb{N}, 0, succ, <, +), X, \Sigma, R\rangle$ be a one-dimensional ICSP. The question of whether P has a solution is undecidable.*

5. FUTURE WORK

We conclude this paper by suggesting some directions for research on ICSP's:

- Having identified decidable classes of ICSP, it would be interesting to find their complexity. It is known that complexity of linear orders with uninterpreted unary predicates is double exponential, which implies that ICSP's with indexing structure $(\mathbb{N}, 0, succ, <)$ can be solved in time $2^{2^{O(s)}}$ where s is the size of the constraint patterns. It would be interesting to identify the subclasses of ICSP's that can be solved in (single) exponential and polynomial time in s.
- We have proven that ICSP's with $(\mathbb{N}, 0, succ, +)$ as indexing structure are undecidable. It would be nice to find general conditions under which the ICSP's are decidable.
- For every ICSP P with indexing structure universe $\mathbb{N}$, there is natural sequence of finite CSP's P_n: P_n is obtained from P by restricting the values of all variables having an index greater than n to some default value ($\perp$ in case of propositional variables). It would be very interesting to find connection between the decidability of P and the complexity of P_n. It would be also nice to prove some kind of compactness theorem saying that if every P_n is satisfiable, then so is P.
- It can also be interesting to consider an extension to our ICSP's to allow constraints whose arity is not fixed. This kind of constraints also arises in practice; e.g., in constraints involving arbitrary summations.
- To a more practical level, given the Büchi automata representation of ICSP's given in this paper, one should look into tools for these automata to see if they can be tailored to deal with ICSP's. A good starting point could be the MONA tool [11].

6. REFERENCES

[1] R. Alur and T. Henzinger. A really temporal logic. In *Proc. 30th IEEE Symp. on Foundations Of Computer Science*, pages 164–169, 1989.

[2] K. Apt. Personal Communication, 2004.

[3] M. Bodirsky and J. Nesetril. Constraint satisfaction with countable homogeneous templates. In *Computer Science Logic*, volume 2803 of *LNCS*, pages 44–57. Springer, 2003.

[4] J. Bowen and D. Bahler. Conditional existence of variables in generalized constraint networks. In *Proc. 9th. National Conference of the American Association for Artificial Intelligence*, pages 215–220, 1991.

[5] J. R. Buchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Cong. on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[6] S. Dantchev and F. Valencia. On infinite csp's. In *Proc. of the CP'04 Workshop on Modelling and Reformulating CSPs*, 2004. Available at www.brics.dk/∼fvalenci.

[7] B. Faltings and S. Macho. Open constraint satisfaction. In LNCS, editor, *Proc. of CP'02*, volume 2470, pages 356–370. Springer-Verlag, 2002.

[8] M. Freedman. k-sat on groups and undecidability. In *Proc. 30th ACM Symp. on Theory Of Computing*, pages 572–576, 1998.

[9] J. Halpern. Presburger arithmetic with unary predicates is π_1^1 complete. *Journal of Symbolic Logic*, 56(2):637–642, 1991.

[10] J. L. Hirst and S. Lempp. Infinite versions of some problems from finite complexity theory. *Notre Dame Journal of Formal Logic*, 37(4):545–553, 1996.

[11] N. Klarlund and A. Møller. *MONA Version 1.4 User Manual*. BRICS Notes Series NS-01-1, Department of Computer Science, University of Aarhus, 2001.

[12] M. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

[13] C. Rueda and F. Valencia. On validity in modelization of musical problems by ccp. In *Formal Systems and Music: Special Issue of Soft Computing*. Springer-Verlag, 2004.

[14] A. Sistla, M. Vardi, and P. Wolper. The complementation problem for buchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.