On Infinite CSP's

Stefan Dantchev and Frank D. Valencia *

[1] Dept. of Mathematics and Computer Science, University of Leicester, UK
[2] Dept. of Information Technology, Uppsala University
Email:dantchev,fvalenci@brics.dk

Abstract We present a new generalization of Constraint Satisfaction Problems (CSP's) to allow *infinitely* (or *unboundedly*) many *indexed* variables. The indices of variables are specified in a first-order decidable theory. We call this generalization Infinite CSP's (ICSP's). Applications of ICSP include problems in which the number of variables is unknown a priori, and optimization problems wrt the number of variables satisfying a given finite set of constraints.
We shall study 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 constrast 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 constrast 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*.

1 Introduction

Constraint satisfaction problems (CSP) occur widely in engineering, science and the arts. Applications are frequently reported in production planning, resource allocation [BLN01], music composition [AADR98], Verification [EM97], Security [BB01] Bioinformatics [GW94] and many others. In fact, a CSP is any problem that can be expressed as that of finding, for a given set of variables, assignments satisfying some given particular properties. These properties are represented by relations called *constraints*.

Motivation. In many CSP's the number of variables may not be not known a priori. This arises in applications such as: scheduling, music generation, and online problems such as auctions. The following quote from [Bes04] exemplifies the importance of this kind of CSP's best :

* The contribution of F. Valencia to this work was supported by the **PROFUNDIS** Project.

"...I remember an application we had to solve some years ago for a company that wanted to make the planning of the employees for two or three years in advance. They knew almost nothing about the constraints on very future days, just that every Friday morning there is the lab meeting, on Wednesday afternoon the children day, etc. Plus some events planned long in advance ... At that time we dreamed of open number of variables, but that didn't exist..." [Bes04].

In this paper we introduce a framework to express CSP's that allows constraints on *unboundedly many indexed variables* but each taking on values from a *finite* domain. We call these problems *Infinite CSP's* (ICSP's). Furthermore, we shall study the decidability of the *satisfiability* (i.e., the existence of solutions) for ICSP's wrt two properties on the indices of the variables: Namely,

- *Index dimension*: E.g., of the form x_i (one dimensional), or $x_{i,j}$ (two dimensional).
- *Language to specify indices*: E.g., using index comparison and summations such as in "$c(x_i, x_{i+j})$ for all j s.t, $j < i$".

Relevance. The study of the decidability of ICSP's satisfiability is relevant for the specification of constraint problems in which one does not know a priori the numbers of variables. A designer can formulate a specification for an arbitrary number of variables by using, for example, constraint expressions of the form $x_i > x_{i+1}$ for $i = 2, 4, 6, \ldots$ or with two-dimensional indexes like in $x_{i,j} = x_{i,j-1}$ for all $i, j > 0$. The satisfiability question, here is of the essence because, after all, as remarked in [Var96], a specification is interesting for a designer only if it is satisfiable.

Another application of decidability studies for ICSP's has to do with *optimization problems* of the kind suggested by [Apt04]. For example, suppose you are asked to provide an (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, if such an m exists. One might suggest an algorithm $\mathcal{A}$ that, unless P is satisfiable for any number of variables, tries incrementally different values of m knowing that eventually it will find the first one after which the problem is no longer satisfiable. Notice that because of the "unless" condition, the decidability of ICSP satisfiability is fundamental here.

Results. The results we present in this paper are the following: (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 ICSP satisfiability is *decidable*. In contrast, (2) for the two-dimensional case, ICSP satisfiability is *undecidable* for indices specified in $(\mathbb{N}, 0, succ)$ and hence also in $(\mathbb{N}, 0, succ, <)$. Furthermore, we prove that (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.

Another interesting aspect of our work is that result (1) is obtained by establishing a connection between ICSP's and a standard computational model:

Namely, Büchi Finite-State Automata [Buc62]. Several theoretical and implementational work, specially in the area of Verification have been carried out for Büchi Automata (see [KM01]). Therefore, we may hope to benefit from this connection both for theoretical and application purposes in the study of ICSP.

Organization. The rest of this paper is organized as follows: Section 2 gives some preliminaries and discusses related work. Section 3 introduces the notion of ICSP. Some examples of ICSP's are given in Section 4. Section 5 presents the above-mentioned decidability results. Finally, some concluding remarks are given in Section 6.

2 Preliminaries and Related Work

2.1 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.

Given a first-order formulae ϕ, we use the notation $\phi(x_1, \ldots, x_n)$ to mean that the free variables of ϕ are in $\{x_1, \ldots, x_n\}$.

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 a countable set.

2.2 Related Work

It should be noticed that the term *infinite CSP* has been used for another different approach: CSP's with infinite *domain* of values (e.g. [OV93]). To our knowledge there is little work on CSP's with infinitely (or unboundedly) many variables. An infinite CSP, and more precisely an infinite SAT, was studied by Freedman [Fre98]. 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 [BN03] 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*.

3 ICSP's

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*, e.g., $x_1, x_2, \ldots$, or $x_{1,1}, x_{1,2}, \ldots$. 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 sub-index 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*),
- $(\{0, \ldots, n-1\}, 0, succ, <, +, \times)$ (i.e., *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 must exist $j < i$ 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 1 (ICSP's). *An ICSP is a tuple* $P = \langle \mathcal{I}, X, \Sigma, R \rangle$ *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 tuples of the form* $\langle G[i_1], \ldots, G[i_n]; c[x_1(s_1), ..., x_n(s_n)] \rangle$*; the* constraint-patterns. *Here* $c \subseteq \Sigma^n$ *is an* n*-ary relation called* constraint, *and for* $k = 1, \ldots, n$*:*
  - $G[i_k] = \mathtt{Q}_{i_k} \triangleright \phi_k$ *with* $\mathtt{Q} \in \{\forall, \exists\}$ *and* $\phi_k(i_1, \ldots, i_k)$ *a formula over* $sig(\mathcal{I})$.
  - $x_k \in X$ *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 1 *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 1. 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 $\mathcal{I} = (\mathbb{N}, 0, succ, <, +)$, i.e., *Presburger Arithmetic*, $X = \{x\}$ 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 $even(i) \overset{\mathtt{def}}{=} \exists_l (l + l = i)$ and $\phi(i, j) \overset{\mathtt{def}}{=} j = i + 1$. □

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 (Solutions & 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*. Such an assignment* v *is a* solution *of* P *if and only* *if for each constraint-pattern* $\langle G[i_1], \ldots, G[i_n]; c[x_1(s_1), ..., x_n(s_n)]\rangle \in R$, *the assignment* v *satisfies, in the structure* $\mathcal{I}_P$, *the formula*

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

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

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

Example 2. The ICSP P in Example 1 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.

4 ICSP's Examples

It is easy to see that many variants of the discrete time temporal logic can be expressed as ICSP's in our setting. Instead of presenting these fairly straightforward constructions, we shall show how to express some musical and well-known recreational mathematics problems 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 [RV04] already 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 first 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 :

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

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.

Infinite Knight Tours. The second problem we express as an ICSP is *the infinite knight tour*. Clearly a knight tour on an infinite chess-board defines a bijection between $\mathbb{Z}^2$ (a square on the chess-board) and $\mathbb{Z}$ (a moment in time), so we will express the fact that there is a such bijection as an ICSP with boolean indexed variables:

$$\langle \forall_{i,j} \exists_u \; ; t_{iju} \rangle$$
$$\langle \forall_{i,j,u,v} \triangleright (u \neq v) \; ; t_{iju} \Rightarrow \neg t_{ijv} \rangle$$
$$\langle \forall_u \exists_{ij} \; ; t_{iju} \rangle$$
$$\langle \forall_{ijzwu} \triangleright (i \neq z) \vee (j \neq w) \; ; t_{iju} \Rightarrow \neg t_{zwu} \rangle$$

The first two lines say that t_{iju} defines a total function from $\mathbb{Z}^2$ to $\mathbb{Z}$. The third and the fourth constraint say that the function is surjective and injective, respectively. For the infinite knight tour, we may think that $t_{iju} = \top$ if and only if the knight is on the square (i, j) at the time u. We then need to add a constraint expressing the fact that if the knight is on the square (i, j) at the time u, then he must jump to one of the neighbouring squares (in the sense of legitimate knight moves) at the time $u+1$. It is not hard to see that the following constraint pattern does the job:

$$\begin{aligned}
&\langle \forall_{i_{-2},i_{-1},i_1.i_2,j_{-2},j_{-1},j_1.j_2,u,v} \triangleright \\
&(i_{-2} = i - 2) \wedge (i_{-1} = i - 1) \wedge (i_1 = i + 1) \wedge (i_2 = i + 2) \wedge \\
&(j_{-2} = j - 2) \wedge (j_{-1} = j - 1) \wedge (j_1 = j + 1) \wedge (j_2 = j + 2) \wedge \\
&(v = u + 1)\,; \\
&t_{iju} \Rightarrow (t_{i_{-1}j_{-2}v} \vee t_{i_{-2}j_{-1}v} \vee t_{i_{-1}j_2v} \vee t_{i_2j_{-1}v} \vee t_{i_1j_{-2}v} \vee \\
&t_{i_{-2}j_1v} \vee t_{i_1j_2v} \vee t_{i_2j_1v}) \rangle
\end{aligned}$$

The underlying indexing structure is $(\mathbb{Z}, succ)$.

5 Decidability Results

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

Convention 2 *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.

5.1 Linear-Order ICSP's: One Dimensional Case

We shall state that one-dimensional ICSP's with linear-order indexing structures can be characterized in terms of Büchi automata [Buc62].

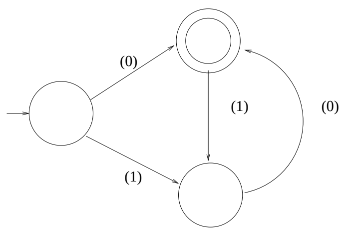
Büchi FSA. Recall that Büchi automata are ordinary finite-state automata (FSA) with an acceptance condition for infinite (or ω) sequences: an ω sequence is accepted iff the automaton can read it from left to right while visiting a final state infinitely often. The language recognized by a Büchi automaton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

We shall use the following notation.

Notation 1 *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 3. Consider a problem P with domain $\Sigma = \{0, 1\}$, indexed variables $x_0, x_1, \ldots$ and constraints $x_i \neq x_{i+1}$ for each $i = 0, 1, 2, \ldots$. Clearly, P can be expressed as a one-dimensional ICSP with indexing structure $(\mathbb{N}, 0, succ, <)$ and set of indexable symbols $X = \{x\}$. Notice that v is a solution of P iff $v(X_0).v(X_1)\ldots$ is accepted by the following Büchi automaton:



□

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

Lemma 1 (Büchi Representation). *Let $P = \langle(\mathbb{N}, 0, succ, <), X, \Sigma, R\rangle$ be 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 (Outline). The construction can be obtained as the Büchi construction given in [Tho90] for Monadic Second-Order Logic formulae interpreted over ω-sequences (S1S). In fact, ICSP's can be translated to S1S: The S1S first-order language has quantifiers over sequence positions which corresponds in our case to indexing variables. Also, S1S includes the predicate $<$ over sequence positions. Furthermore, for every symbol a in the signature of sequences, S1S has a predicate $a?(i)$ to test whether the symbol a occurs at position i of the interpreting sequence. Since constraints are finite relations such predicates can be used to represent them using disjunctive normal form in the obvious way. □

As a corollary of Lemma 1 we get the decidability of satisfiability for the ICSP's under consideration.

Theorem 1. *Given a one-dimensional ICSP $P = \langle(\mathbb{N}, 0, succ, <), X, \Sigma, R\rangle$, the question of whether P has a solution is decidable.*

Proof. It follows from Lemma 1 and the decidability of the emptiness problem for Büchi automata [SVW87]; whether a given Büchi automaton accepts at least one sequence. □

Relevance of the Büchi FSA Representation. Let us discuss a little about the relevance of Lemma 1. First, Büchi automata give us a finite representation of all the solutions of a given one-dimensional ICSP's with linear-orders. Second, (the languages of) Büchi automata are closed under intersection, disjunction and complementation. This allows to perform compositional analysis of ICSP's. So, for example, if one is given two automata B_P and B_Q representing the solutions of two ICSP' P and Q (possibly with common variables and domains), then one can perform an operation on these two automata, to get an automata representing the (solutions of the) conjunction between P and Q.

5.2 Linear-Order ICSP's: Two-Dimensional Case

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 specification expressiveness of linear-order one-dimensional ICSP which from Lemma 1 does not go beyond of that of Büchi FSA.

Our reduction lemma can be stated as follows:

Lemma 2 (Turing Reduction). *Let M be a deterministic Turing machine. One can effectively construct a two-dimensional ICSP P_M with indexing structure $(\mathbb{N}, 0, succ)$ such that:*

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

Proof (Outline). Given M, our ICSP P_M has a variable $x_{i,t}$ for $i, t = 0, 1, 2, ...$ representing the state of the Turing machine cell i at time t including the state of the finite control if the head is at that cell.

Furthermore, the ICSP P_M has constraint σ on $x_{i,t}$, $x_{i,t-1}$, and $x_{i,t}$'s adjacent cells $x_{i-1,t-1}$ and $x_{i+1,t-1}$ at time $t-1$. For every $i \geq 0$ and $t > 0$, the constraint $\sigma(x_{i,t}, x_{i-1,t-1}, x_{i,t-1}, x_{i+1,t-1})$ specifies how the cell i at time i is computed as a transition from the configuration at time $t-1$; and if M is in the halting state at time $t-1$ with the head on one of $i-1$, i or $i+1$, no value of $x_{i,t}$ should satisfy the constraint. Finally, P_M has also constraint σ_0 describing the initial configuration.

One can verify that P_M can be specified as a two-dimensional ICSP with indexing structure $(\mathbb{N}, 0, succ)$ and that it has a solution if and only if the machine does not halt. $\square$

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, also two-dimensional ICSP's with linear-order as indexing structure are undecidable.

Theorem 2. *Given a two-dimensional ICSP $P = \langle \mathcal{I}, X, \Sigma, R \rangle$ with indexing structure $\mathcal{I} \in \{(\mathbb{N}, 0, succ), (\mathbb{N}, 0, succ, <)\}$, the question of whether P has a solution is undecidable.*

5.3 Presburger ICSP's: One Dimensional Case

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) [Hal91]. It is also known that certain temporal logics based on Presburger Arithmetic are undecidable, too [AH89]. Unfortunately neither proof applies to our setting.

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. They were introduced by Minsky in [Min61] and are sometimes called Minsky machines.

Minsky two-counter machines. Recall that a two-counter machine is defined as follows. 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.

As in the case of two-dimensional linear-order ICSP's, the reduction also states that Presburger ICSP's are very expressive as they are capable of specifying arbitrary Turing computations. However, in Presburger this already happens for one-dimensional ICSP's.

We can now state the reduction lemma:

Lemma 3 (Minsky Reduction). *Let $\mathcal{M}$ be a two-counter machine with a sequence of m instructions $\alpha_0, \alpha_1, \ldots \alpha_{m-1}$ and a natural number n. One can effectively construct a one-dimensional ICSP $P_{\mathcal{M}}$ with an indexing structure $(\mathbb{N}, 0, succ, <, +)$ such that:*

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

The proof of the above lemma is more involved than that of Lemma 2. So, we will devote the end of this section to describe it. But first, we state the undecidability of Presburger one-dimensional ICSP's which follows immediately from Lemma 3 and the undecidability of the Halting problem for two-counter machines.

Theorem 3. *Given a one-dimensional $P = \langle (\mathbb{N}, 0, succ, <, +), X, \Sigma, R \rangle$, the question of whether P has a solution is undecidable.*

Proof of Lemma 3. Let $\mathcal{M}, n$ as in Lemma 3. We describe the construction of a one-dimensional ICSP $P_{\mathcal{M}}$ with indexing structure $(\mathbb{Z}, 0, succ, <, +)$ and indexed boolean variables $\{p_u \mid u \in \mathbb{Z}\}$ s.t.,

$$P_{\mathcal{M}} \text{ is satisfiable if and only if } \mathcal{M} \text{ does not halt on the input} \tag{1}$$

The proof of Lemma 3 can be obtained via the standard isomorphism between $\mathbb{Z}$ and $\mathbb{N}$ as it can be expressed with the operations from $(\mathbb{N}, 0, succ, <, +)$.

Let us first describe the construction intuitively. We encode the states of $\mathcal{M}$ by positive integers. We use the variables $\{p_u \mid u \in \mathbb{Z}\}$ of the ICSP $P_{\mathcal{M}}$ to encode a property $\mathcal{P}$ on positive integers. The constraints of $P_{\mathcal{M}}$ specify that: $\mathcal{P}$ holds at the initial state of $\mathcal{M}$; if $\mathcal{P}$ hold at any state then it holds at its successor; $\mathcal{P}$ does not hold at any possible final state of $\mathcal{M}$. This implies the statement in Equation 1.

The above-mentioned constraints can be expressed by *purely universal constraints*, so we simplify the notation further: since each generator is of the form $\mathsf{Q}_{x_1,x_2,\ldots x_k} \triangleright \varphi(x_1, x_2, \ldots x_k)$, we omit $\mathsf{Q}_{x_1,x_2,\ldots x_k}\triangleright$-part and simply write $\varphi(x_1, x_2, \ldots x_k)$. Also, notice that we are using $x_1, x_2, \ldots$ to denote indices (or *indexing* variables); the *indexed* variables of the ICSP will be denoted by variables of the form p_u,p_x, p_y, etc.

We now define the encodings. We use the negative indices to encode the powers of two, i.e.

$$\text{for every } u < 0 \; p_u = \top \text{ if and only if } -u = 2^v \text{ for some } v \geq 0.$$

We will need these in order to check the correctness of the value in c_1 when the machine halts at α_{m-1}. This condition can be expressed by the following set of constraint-patterns:

$$\begin{aligned} &\langle x = -1 \; ; \, p_x \rangle \\ \langle (x < -1) \wedge \neg \exists u \, (x = u + u) \; &; \, \neg p_x \rangle \\ \langle (x < 0) \wedge (x = y + y) \; &; \, p_x \Leftrightarrow p_y \rangle. \end{aligned}$$

The first constraint says that 1 is a power of two; the second one says that no odd number is a power of two; the third one says that an even number x is a power of two if and only if $x/2$ is.

We use positive indices to encode the *states of the machine* $\mathcal{M}$, each state consisting of the index i of the current instruction α_i $(0 \leq i < m)$ together with the content of the counters c_1 and c_2. We encode such a triple by a single natural number $x = i + m2^{c_1}3^{c_2}$. Clearly the initial state is encoded by $m2^n$. The fact that$\mathcal{P}$ holds at the initial state of $\mathcal{M}$ is expressed by the following simple constraint-pattern:

$$\langle x = m2^n \; ; \, p_x \rangle$$

(note that $m2^n$ is a given constant, so that $x = m2^n$ is a legitimate formula in Presburger Arithmetic).

In order to say that if $\mathcal{P}$ holds at the state x then it holds at the successor state y, we first need to "decode" x. The instruction index can be obtained by taking x mod m; the increment of c_1 (c_2) can be done by multiplying x div m by 2 (respectively 3); the decrement of of c_1 (c_2) can be done by dividing x div m by 2 (respectively 3) if it is greater than zero, i.e. if x div m is divisible by 2 (respectively 3). We shall use the following abbreviation:

$$DIV_m(x, u, w) \text{ which stands for } \left(x = \underbrace{u + u + \cdots u}_{m \text{ times}} + w \right) \wedge (w \geq 0) \wedge (w < m);$$

the intended meaning is that the quotient and the remainder when x is divided by (the constant) m are u and w, respectively.

Depending on the instruction α_i, we have one of the following four possibilities for every i, $0 \leq i \leq m-2$:

1. α_i : add 1 to c_1 and go to α_j. The current state is encoded by $i + m2^{c_1}3^{c_2}$ for some $c_1, c_2 \in \mathbb{N}$ and the successor state is $j + m2^{c_1+1}3^{c_2}$:

$$\langle (x > 0) \wedge \exists u \, (DIV_m \, (x, u, i) \wedge DIV_m \, (y, u + u, j)) \; ; p_x \Rightarrow p_y \rangle.$$

2. α_i : add 1 to c_2 and go to α_j. This is pretty similar to the previous case:

$$\langle (x > 0) \wedge \exists u \, (DIV_m \, (x, u, i) \wedge DIV_m \, (y, u + u + u, j)) \; ; p_x \Rightarrow p_y \rangle.$$

3. α_i : if $c_1 = 0$ go to α_j else subtract 1 from c_1 and go to α_k. The current state is encoded by $i + m2^{c_1}3^{c_2}$ for some $c_1, c_2 \in \mathbb{N}$. If $c_1 = 0$ then the successor state is $j + m2^{c_1}3^{c_2}$. If $c_1 > 0$ then the successor state is $k + m2^{c_1-1}3^{c_2}$.

$$\begin{aligned} \langle (x > 0) \wedge DIV_m \, (x, u, i) \wedge (\neg \exists w \, (u = w + w) \wedge DIV_m \, (y, u, j) \\ \vee \exists w \, ((u = w + w) \wedge DIV_m \, (y, w, k))) \; ; p_x \Rightarrow p_y \rangle \end{aligned}$$

4. α_i : if $c_2 = 0$ go to α_j else subtract 1 from c_2 and go to α_k. This is similar to the previous case:

$$\begin{aligned} \langle (x > 0) \wedge DIV_m \, (x, u, i) \wedge (\neg \exists w \, (u = w + w + w) \wedge DIV_m \, (y, u, j) \\ \vee \exists w \, ((u = w + w + w) \wedge DIV_m \, (y, w, k))) \; ; p_x \Rightarrow p_y \rangle \end{aligned}$$

Finally we need to say that $\mathcal{P}$ does not hold at any possible final state, i.e. a state of the form $m - 1 + m2^p$ for some $p \in \mathbb{N}$:

$$\langle \exists u \, ((y < 0) \wedge (u + y = 0) \wedge DIV_m \, (x, u, m-1)) \; ; p_y \Rightarrow \neg p_x \rangle$$

(note that here $y < 0$, so that p_y has nothing to do with the property $\mathcal{P}$ defined on the states of $\mathcal{M}$, but simply says that $-y$ is a power of two). $\square$

6 Concluding Remarks & Future Work

We have presented ICSP's; a framework to specify CSP's with unbounded number of indexed variables. We proved decidability of one-dimensional ICSP's with linear-order indexing structures. We also proved undecidability for two-dimensional ICSP's with linear-order indexing structures as well as for one-dimensional ICSP's with Presburger indexing structures. The decidability result was obtained by a reduction to Büchi automata while the undecidability ones were obtained by a reduction from Turing and Minsky machines. We argued that the Büchi representation of one-dimensional ICSP's with linear-order may help to provide for the compositional analysis of these problems.

We conclude this paper by suggesting some theoretical and practical direction 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 with indexing structure $(\mathbb{N}, 0, succ, +)$ is 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 [KM01].

Acknowledgments

We are indebted to Krzysztof Apt, Christian Bessière, Rina Dechter, Camilo Rueda and Pascal Van Hentenryck for indicating to us relevant aspects of ICSP's.

We also wish to thank Peter Bro Miltersen and Parosh Abdula for technical discussions on the topics of this paper.

References

[AADR98] C. Agon, G. Assayag, O. Delerue, and C. Rueda. Objects, time and constraints in openmusic. In *ICMC98*, pages 1–12. ICMA, 1998.

[AH89] R. Alur and T.A. Henzinger. A really temporal logic. In *Proc. 30th IEEE Symp. on Foundations Of Computer Science*, pages 164–169, 1989.

[Apt04] K.R. Apt. Personal Communication, 2004.

[BB01] G. Bella and S. Bistarelli. Soft constraints for security protocol analysis: Confidentiality. *LNCS*, 1990, 2001.

[Bes04] C. Bessière. Personal Communication, 2004.

[BLN01] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling. Applying Constraint programming to Scheduling Problems.* Kluwer, 2001.

[BN03] M. Bodirsky and J. Nesetril. Constraint satisfaction with countable homogeneous templates. In M. Baaz and J.A. Makowsky, editors, *Computer Science Logic*, volume 2803 of *LNCS*, pages 44–57. Springer, August 2003.

[Buc62] 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.

[EM97] J. Esparza and S. Melzer. Model checking LTL using constraint programming. In *18th International Conference on Application and Theory of Petri Nets*, volume 1248, pages 1–20. Springer-Verlag, 1997.

[Fre98] M. Freedman. k-sat on groups and undecidability. In *Proc. 30th ACM Symp. on Theory Of Computing*, pages 572–576, 1998.

[GW94] C. Gaspin and E. Westhof. The determination of secondary structures of RNA as a constraint satisfaction problem. In *Advances in molecular bioinformatics*. IOS Press, 1994.

[Hal91] J.Y. Halpern. Presburger arithmetic with unary predicates is π_1^1 complete. *Journal of Symbolic Logic*, 56(2):637–642, June 1991.

[KM01] N. Klarlund and A. Møller. *MONA Version 1.4 User Manual*. BRICS Notes Series NS-01-1, Department of Computer Science, University of Aarhus, January 2001.

[Min61] 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, November 1961.

[OV93] W. Older and A. Vellino. Constraint Arithmetic on Real Intervals. In Frédéric Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming: Selected Research*. MIT Press, 1993.

[RV04] C. Rueda and F. Valencia. On validity in modelization of musical problems by ccp. In *Formal Systems and Music: Special Issue of Soft Computin*. Springer-Verlag, 2004.

[SVW87] 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.

[Tho90] W. Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, Formal models and semantics, pages 133–191. Elsevier, 1990.

[Var96] M. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, volume 1043 of *LNCS*, pages 238–266. Springer, 1996.