The complexity of recursive constraint satisfaction problems.

Victor W. Marek
Department of Computer Science
University of Kentucky
Lexington, KY 40506, USA
marek@cs.uky.edu

Jeffrey B. Remmel
Department of Mathematics
University of California
La Jolla, CA 92093, USA
jremmel@ucsd.edu

January 31, 2008

Abstract

We investigate the complexity of finding solutions to infinite recursive constraint satisfaction problems. We show that, in general, the problem of finding a solution to an infinite recursive constraint satisfaction problem is equivalent to the problem of finding an infinite path through a recursive tree. We also identify natural classes of infinite recursive constraint problems where the problem of finding a solution to the infinite recursive constraint problem is equivalent of the problem of finding an infinite path through a recursive finitely branching recursive trees or a recursive binary tree. There are a large number of results in the literature on the complexity of the problem of finding an infinite path through a recursive tree. Our main result allows us to automatically transfer such results to to give equivalent results about the complexity of the problem of finding a solution to a recursive CSP problem.

1 Introduction

Constraint Programming and, more specifically, Constraint Satisfaction Problems (CSP) is a declarative paradigm normally used to describe search problems by means of the *constraints* that *solutions* to the problem must satisfy. As such, CSP has a very long history, essentially reaching to the beginning of mathematics, since we can think of the problem of finding solutions to equations or systems of equations as instances of constraint satisfaction problems. In the modern way of thinking about

CSPs, the constraints limit the acceptable values of *variables*. As such, constraints are often represented explicitly as *tables*. For example, the constraint on two non-negative integer values x and y which satisfy $x + y = 7$ is, in fact, a table consisting of 8 values: $(0, 7), (1, 6)$, etc. When a collection $\mathcal{C}$ of such constraints is given, we are seeking an assignment to variables v so that for every constraint C in $\mathcal{C}$, the restriction of v to variables of C is a row in C. Thus, we can think of constraints as "local" restrictions on the set of acceptable solutions and our goal in a given constraint satisfaction problem is to find a solution that simultaneously satisfies all such local restrictions.

The issue of representation of constraints, i.e. the language that is used to represent them, is usually not the subject of theory of CSP. However, each specific domain has its own language. We refer the reader to the recent monographs [Apt03, Dec03] for the description of the general theory of CSPs. Of course, for specific types of CSP problems such a finding solutions to diophantine equations, linear programming problems, or integer programming problems, the techniques developed for finding solutions depends heavily on the particular representation of the problem. It is only when we abstract from such particular representations that we can talk about representing all such constraints as tables. We note that one popular knowledge representation language that is often used to represent all finite CSPs is where the set of variables is consider to be the set of variables of propositional logic and where the constraints are represented by *propositional clauses*. That is, we *encode* constraints as collections of Boolean clauses. Here, for a given CSP problem P, i.e. a finite collection of finite tables, we represent the problems as a finite CNF formula φ_P in such a way that there is a one-to-one correspondence between satisfying valuations for φ_P and solutions to P. This translation is *modular*, i.e. adding additional constraints results in larger CNF formula. Yet another formalism that has the same property is the Answer Set Programming [MT99, Nie99] which also allows for a faithful modular translation. Other formalisms that have similar properties include 0-1 Integer Programming and its variation where the variables are only allowed to take -1 and 1 as values [Hoo00, Sch98]. One can devise many other formalisms that allow for faithful translation of CSPs. Each of these formalisms has its specific constraint manipulation rules. For example, the CNF representation uses Boolean constraint propagation to manipulate formulas while integer programming allows for use of cutting plane rules. Of course, the abstract version of CSP itself has its own rules such as *arc-consistency* and its variations, see [Apt03] for a proof-theoretic presentation of this and other techniques. In general, such techniques do not necessarily directly translate from one formalism to another. Hence, it is possible that translating a CSP from one formalism to another may effect the complexity of finding solutions, see [CCT87].

The goal of this paper is to define a natural class of infinite constraint satisfaction problems in such a way that we can use the tools of modern recursion theory to study the complexity of finding solutions. In particular, we define and classify *infinite re-*

cursive constraint satisfaction problems. In general, we shall show that the problem of finding a solution to an infinite recursive constraint satisfaction problem is equivalent to the problem of finding an infinite path through an infinite recursive tree. That is, we shall show that for any infinite recursive CSP problem P, there is a recursive tree T_P such that there is a one-to-one degree preserving correspondence between the set of solutions to P and the set of infinite paths through T_P and, vice versa, for any infinite recursive tree T, there is an infinite recursive CSP problem P_T such that there is is a one-to-one degree preserving correspondence between the set of infinite paths through T and the set of solutions to P_T. It will follow that the problem of finding a solution to an infinite recursive CSP problem is a Σ_1^1 complete problem. We define natural classes of infinite recursive CSP problems called *finitely based* and *bounded* CSPs where the problem of finding solutions is equivalent to the problem of finding infinite paths through finitely branching recursive trees and, hence, the complexity of the problem of finding solutions for such CSPs is greatly reduced. We also define a class of infinite recursive CSPs called *recursively bounded* CSPs where the problem of finding solutions is equivalent to the problem of finding an infinite path through a binary recursive tree. There is an extensive literature on the complexity of the problem of finding infinite paths through various types of recursive trees, see [JS72a, JS72b, JLR91] for example. Our basic coding result will allow us to automatically transfer such results to give corresponding results on the complexity the problem of finding solutions to recursive CSPs.

The outline of this paper is as follows, In section 2, we shall define the various classes of infinite recursive CSPs described above and give the required codings to show that the problem of finding solutions to these infinite recursive CSPs is equivalent to the problem of finding infinite paths through recursive trees. In section 3, we shall use the results of section 2 to derive various index set results for for CSPs that possess at least one solution, no solutions, finitely many solutions, or infinitely many solutions. Similar, we can prove index set results on classes of CSPs which possess a recursive solution or no recursive solution. In section 4, we give a number of results about the degrees of solutions to infinite recursive CSPs that can be derived by transferring results on the degrees of infinite paths through recursive trees.

2 Constraint Satisfaction Problems and Trees

As we described in the introduction, the goal of this paper is to show that there is a close connection between the problems of finding solutions to recursive constraint satisfaction problems and finding paths through recursive trees.

First we need to define constraint satisfaction problems.

Definition 2.1. *For any set A and any natural number n, the set of all n-tuples of* *elements of A is denoted by A^n. Any subset of A^n is called an n-ary relation over A. The set of all finitary relations over A is denoted by R_A. A* **constraint language over** *A is a subset of R_A.*

Definition 2.2. *For any set A and constraint language Γ over A, the constraint satisfaction problem* $\mathbf{CSP}(\Gamma, A)$ *is the following combinatorial problem.*

Problem Instances *are triples (V, A, C) where*

1. *V is a set of variables,*
2. *$C = \{C_i : i \in \Omega\}$ is a set of constraints where Ω is some indexing set and each constraint C_i is a pair $\langle s_i, \rho_i \rangle$ where s_i is a tuple of variables from V of length m_i called the* **constraint scope** *and $\rho_i \in \Gamma$ is an m_i-ary relation called the* **constraint relation***, and*
3. *for each variable $v_i \in V$, there is a constraint $C_j \in C$ such that v_i occurs in the tuple s_j.*

Solutions: *A solution to (V, A, C) is a function $\phi : V \to A$ such that for each constraint $C_i = \langle s_i, \rho_i \rangle$ with $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$, the tuple $(\phi(v_{j_1}), \ldots, \phi(v_{j_{m_i}}))$ is in ρ_i.*

We let $\mathcal{S}(V, A, C)$ denote the set of all solutions of the constraint satisfaction problem (V, A, C).

Before we can define recursive constraint satisfaction problems, we must first establish some basic notation from recursion theory. Let ω denote the set of natural numbers $\{0, 1, \ldots\}$ Let $[,] \colon \omega \times \omega \to \omega$ be a fixed one-to-one and onto recursive pairing function such that the projection functions π_1 and π_2 defined by $\pi_1([x, y]) = x$ and $\pi_2([x, y]) = y$ are also recursive. Let $\omega^{<\omega}$ denote the set of all finite sequences from ω and let $2^{<\omega}$ denote the set of all finite sequences of 0's and 1's. Given $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \ldots, \beta_k \rangle$ in $\omega^{<\omega}$, we write $\alpha \sqsubseteq \beta$ if α is initial segment of β, i.e., if $n \leq k$ and $\alpha_i = \beta_i$ for $i \leq n$. For the rest of this paper, we shall identify a finite sequence $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ with its code $c(\alpha) = [n, [\alpha_1, \ldots, \alpha_n]]$ in ω. We assume that 0 is the code of the empty sequence $\emptyset$. Thus, when we say that a set $S \subseteq \omega^{<\omega}$ is recursive (recursively enumerable, etc.), we will mean that the set $\{c(\alpha) \colon \alpha \in S\}$ is recursive, (recursively enumerable, etc.) Given a finite set A, we let canonical index of A, $can(A)$, be 0 if A is empty and be $2^{x_1} + \cdots + 2^{x_k}$ if $A = \{x_1 < \cdots < x_k\}$.

A *tree* T is a nonempty subset of $\omega^{<\omega}$ closed under initial segments. We shall identify a tree T contained in $\omega^{<\omega}$ with the set of codes of the nodes in T. Thus we think of T as a certain subset of ω. With this convention, a tree T contained in $\omega^{<\omega}$ is *recursive* if the set of codes of nodes in T is a recursive subset of ω. If T is a tree contained in $\omega^{<\omega}$, then a function $f\colon \omega \to \omega$ is called an infinite *path* through T if for all n, $\langle f(0), \ldots, f(n)\rangle \in T$. Let $[T]$ denote the set of all infinite paths through T. We shall say that T is finitely branching if there is a function $f : T \to \omega$ such that for all nodes $\eta = (\eta_1, \ldots, \eta_n) \in T$, $(\eta_1, \ldots, \eta_n, j) \in T$ implies $j \leq f(\eta)$. If T is recursive tree and f is partial recursive function, then we say that T is **highly recursive**. It is easy to see that a recursive tree T is highly recursive if and only if T is finitely branching and there is an effective procedure which for each node $\eta \in T$ produces the canonical index of the set of all immediate successors of $\eta \in T$.

A set A of functions is called a Π^0_1-class if there is a recursive predicate R such that $A = \{f\colon \omega \to \omega : \forall_n (R([f(0), \ldots, f(n)])\}$. It is well known that for each Π^0_1-class C, there is a recursive tree T_C such that $C = [T_C]$ and that for any recursive tree T, $[T]$ is Π^0_1-class. Thus, we shall always think of a Π^0_1-class as the set of paths through a recursive tree $T \subseteq \omega^{<\omega}$. Note that if T is a tree contained in $2^{<\omega}$, then $[T]$ is a collection of $\{0,1\}$-valued functions and we can identify each $f \in [T]$ with the set A_f, $A_f = \{x\colon f(x) = 1\}$. Thus, in such a case, we can think of $[T]$ as a Π^0_1 class of *sets.* We say that C is a **bounded** Π^0_1 class if $C = [T]$ for some finitely branching recursive tree T and we say C is a **recursively bounded** Π^0_1 class if $C = [T]$ for some highly recursive tree T.

If A is a recursive set, then we say that a relation $R \subseteq A^n$ is recursive if the set of codes $c(\vec{a})$ such that $\vec{a} = (a_0, \ldots, a_{n-1}) \in R$ is a recursive set. Let ϕ_e denote the partial recursive function computed by the e-th Turing machine so that $\phi_0, \phi_1, \ldots$ is an effective list of all partial recursive functions. Given any oracle A, we let ϕ^A_e denote the partial recursive function computed by the e-th oracle Turing machine with oracle A so that $\phi^A_0, \phi^A_1, \ldots$ is an effective list of all A-partial recursive functions. We say that e is a recursive index of the recursive set $A \subseteq \omega$ if ϕ_e is the characteristic function of A. Thus an index of a recursive tree $T \subseteq \omega^{<\omega}$ is just an index of the recursive set consisting of all codes of nodes in T.

Definition 2.3. *A constraint satisfaction problem* (V, A, C) *is said to be an* **infinite recursive** *constraint satisfaction problem if the following hold.*

1. $V \subseteq \{v_i :\in \omega\}$ *is a recursive set of variables, i.e. if* $\{i : v_i \in V\}$ *is a recursive set.*
2. A *is a recursive set of natural numbers,*
3. C *is an infinite effective sequence of recursive constraints* $\{C_i : i \in \Omega\}$ *where* Ω *is some infinite recursive subset of the natural numbers. That is, each constraint* C_i *is a pair* $\langle s_i, \rho_i\rangle$ *where* $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ *is a tuple of variables from* V *of length* m_i *called the* **constraint scope**, $\rho_i \in \Gamma$ *is a recursive* m_i*-array relation called the* **constraint relation**, *and there is a partial recursive function* f *is that*

for each $i \in \Omega$, $f(i) = [m_i, a_i, b_i]$ where a_i is a code of the m_i-tuple $(j_1, \ldots, j_{m_i})$ and b_i is a recursive index of ρ_i.

4. *For each variable $v_i \in V$, there is a constraint $C_j = \langle s_j, \rho_j \rangle$ such that v_i occurs in s_j.*

Let us observe that there is no loss in generality in assuming that $V = \{v_i :\in \omega\}$ and that $\Omega = \omega$.

We now introduce a classification of recursive constraint satisfaction problems. We say that an infinite recursive constraint problem (V, A, C) is

1. **finitely based** if A is finite,
2. **bounded** if for each constraint $C_i = \langle s_i, \rho_i \rangle$, ρ_i is finite, and
3. **recursively bounded** if either it is finitely based or there is a recursive function g such that for all $i \in \omega$ and each constraint $C_i = \langle s_i, \rho_i \rangle$, $\rho_i \subseteq \{0, \ldots, g(i)\}^{m_i}$.

We say that there is an effective one-to-one degree preserving correspondence between the set of solutions $\mathcal{S}(P)$ of a recursive constraint satisfaction problem $P = (V, A, C)$ and the set of infinite paths $[T]$ through a recursive tree T if there are indices e_1 and e_2 of oracle Turing machines such that
(i) $\forall_{\pi \in [T]} \phi_{e_1}^{gr(f)} = f_\pi \in \mathcal{S}(P)$,
(ii) $\forall_{f \in \mathcal{S}(P)} \phi_{e_2}^{gr(f)} = \pi_f \in [T]$, and
(iii) $\forall_{\pi \in [T]} \forall_{f \in \mathcal{S}(P)} (\phi_{e_1}^{gr(\pi)} = f_\pi$ if and only if $\phi_{e_2}^{gr(f_\pi)} = \pi)$.
Here if f is a function $f \colon \omega \to \omega$, then $gr(f) = \{[x, f(x)] \colon x \in \omega\}$. Condition (i) says that infinite paths through tree T, *uniformly* produce solutions to the constraint satisfaction problem (V, A, C) via an algorithm with index e_1. Condition (ii) says that solutions to the constraint satisfaction problem (V, A, C) uniformly produce paths through the tree T via an algorithm with index e_2. We say that A is *Turing reducible* to B, written $A \leq_T B$, if $\phi_e^A = B$ for some e. A is *Turing equivalent* to B, written $A \equiv_T B$, if both $A \leq_T B$ and $B \leq_T A$. Thus condition (iii) asserts that our correspondence is one-to-one and if $\phi_{e_1}^{gr(\pi)} = f_\pi$, then f_π is Turing equivalent to π. In what follows we will not explicitly construct indices e_1 and e_2, but it will be clear that such indices exist in each case.

Theorem 2.4. *Suppose that $P = (V, A, C)$ is a finitely based infinite recursive constraint problem such that $V = \{v_i : i \in \omega\}$ and $C = \{C_j : j \in \omega\}$. Then there is a highly recursive tree T such that there is an effective one-to-one correspondence between the set $\mathcal{S}(P)$ of solutions to the constraint satisfaction P and the set $[T]$ of all infinite paths through T.*

Proof: Let $C_i = \langle s_i, \rho_i \rangle$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$. Let f be the partial recursive function such that for each $i \in \omega$, $f(i) = [m_i, a_i, b_i]$ where a_i is a code of the m_i-tuple $(j_1, \ldots, j_{m_i})$ and b_i is a recursive index of ρ_i. Then it is easy to construct T. Specifically, we put the empty sequence $\emptyset$ in T and we put a node $\eta = (\eta_1, \ldots, \eta_n)$ if and only if
(1) $\eta_j \in A$ for $j = 1, \ldots, n$ and
(2) for all $i \leq n$, if the variables in $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ are contained in $\{v_i : i < n\}$ and we define $\phi(v_i) = \eta_{i+1}$ for $i = 0, \ldots, n-1$, then $(\phi(v_{j_1}), \ldots, \phi(v_{j_{m_i}})) \in \rho_i$.

It is easy to see that T is highly recursive tree and that $\phi \in [T]$ if and only if for all i, if $C_i = \langle s_i, \rho_i \rangle$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$, then $(\phi(v_{j_1}), \ldots, \phi(v_{j_{m_i}})) \in \rho_i$. Thus in this case, $\mathcal{S}(P) = [T]$. $\square$

Before preceding with the other results of this section, we pause to make two comments about the construction in Theorem 2.4. First, we can apply the construction to any finitely based infinite CSP problem whether it is recursive or not. This shows that we can reduce the problem of finding a solution to P to the problem of finding an infinite path through the tree T described in the proof Theorem 2.4. One consequence of this is the following purely combinatorial corollary.

Corollary 2.5. *Suppose that $P = (V, A, C)$ is a finitely based infinite constraint problem such that $V = \{v_i : i \in \omega\}$ and $C = \{C_j : j \in \omega\}$. Then suppose that for any finite set $S \subseteq \omega$, there is a solution to the finite constraint problem $P_S = (V_S, A, C_S)$ where $C_S = \{C_i : i \in S\}$ and $V_S = \{v_i : \exists j \in S(v_i$ occurs in $s_j\}$. Then P has a solution.*

Proof: For each i, let $S_n = \{0, 1, \ldots, n-1\}$. It is easy to see that the fact that finite constraint problem P_{S_n} has a solution means that there will be a node $(\eta_1, \ldots, \eta_n)$ in the tree T constructed in Theorem 2.4. This means that T is a finitely branching infinite tree and, hence, by König's Lemma, T must have an infinite path. Thus as $\mathcal{S}(P) = [T]$, it follows that P has a solution. $\square$

In fact, the construction of Theorem 2.4 works for any infinite recursive constraint satisfaction problem. The only problem with this construction is that the resulting tree T may be infinitely branching even if $P = (V, A, C)$ is bounded or recursively bounded. Our next result will use a slightly different construction of the desired tree T which will have the property that T will be finitely branching if (V, A, C) is bounded and T will be highly recursive if T is recursively bounded.

Theorem 2.6. *1. Let $P = (V, A, C)$ be an infinite recursive constraint satisfaction problem where $V = \{v_i : i \in \omega\}$ and $C = \{C_i = \langle s_i, \rho_i \rangle : i \in \omega\}$. Then there exists a recursive tree $T \subseteq \omega^{<\omega}$ and an effective one-to-one degree preserving correspondence between the set $\mathcal{S}(P)$ of solutions to the constraint satisfaction P and the set $[T]$ of all infinite paths through T.*

2. *If, in addition, $P = (V, A, C)$ is bounded, then T is bounded.*

3. *If, in addition, $P = (V, A, C)$ is recursively bounded, then T is highly recursive.*

Proof: Let $C_i = \langle s_i, \rho_i \rangle$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$. Let f be the partial recursive function such that for each $i \in \omega$, $f(i) = [m_i, a_i, b_i]$ where a_i is a code of the m_i-tuple $(j_1, \ldots, j_{m_i})$ and b_i is a recursive index of ρ_i. For each $i \in \omega$, let $\max(s_i)$ be the largest k such that v_k occurs in the m_i-tuple s_i. Let the domain of ρ_i, $dom(\rho_i)$, equal the set of all $x \in A$, such that there exists an m_i-tuple $\vec{a} \in \rho_i$ such that x occurs in $\vec{a}$. Since the characteristic function of ρ_i is given by the recursive function ϕ_{b_i}, it is easy to see that there is a recursive function F such that $W_{F(i)} = dom(\rho_i)$ where $W_e =$ the domain of ϕ_e is the e-th r.e. set. Since we are assuming that for each i, there is a j such that v_i occurs in s_j, then there is a recursive function G such that $G(i)$ is the least j such that v_i occurs in s_j. Hence, there is a recursive function H such that $W_{H(i)} = dom(\rho_{G(i)})$.

For any e, we let $W_{e,s}$ denote the set of elements $x \leq s$ such that $\phi_e(x)$ converges in s or fewer steps. By convention, we let $W_{e,-1} = \emptyset$. For any sequence, $\eta = (\eta_1, \ldots, \eta_{2n})$ in $\omega^{<\omega}$, we define the map $\psi_\eta : \{v_0, \ldots, v_{n-1}\} \to A$ by $\psi_\eta(v_j) = \eta_{2(j+1)}$ for $j = 0, \ldots, n-1$.

First we put the empty sequence $\emptyset$ into T. Next we will put a node $\eta = (\eta_1, \ldots, \eta_{2n})$ into T if and only if the following conditions hold:

1. $\eta_{2j} \in A$ for $j = 1, \ldots, n$,

2. for $j = 1, \ldots, n$, η_{2j-1} is the least s such that $\eta_{2i} \in W_{H(i-1),s}$.

3. for all $i < n$ such that $\max(s_i) < n$ and $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$, it is the case that $(\psi_\eta(v_{j_1}), \ldots, \psi_\eta(v_{j_{m_i}})) \in \rho_i$.

Thus we put a node $\eta = (\eta_1, \ldots, \eta_{2n})$ into T if and only the map ψ_η gives a solution to all constraints C_i such that $i < n$ and the variables mentioned in C_i come from $\{v_0, \ldots, v_{n-1}\}$. Finally, we put a node $\eta = (\eta_1, \ldots, \eta_{2n}, \eta_{2n+1})$ into T if and only if

1. $(\eta_1, \ldots, \eta_{2n})$ meets the conditions to be put into T and

2. $W_{H(n),\eta_{2n+1}} - W_{H(n),\eta_{2n+1}-1} \neq \emptyset$.

It is easy to see that T is a recursive tree. Now suppose that $\pi = (\pi_1, \pi_2, \ldots)$ is an infinite path through T. Let f_π be the function such that $f_\pi(i) = \pi_{2i+2}$. Then for any constraint $C_i = \langle s_i, \rho_i \rangle$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$, let $t = \max(s_i)$. Then since the node $(\eta_1, \ldots, \eta_{2t+2})$ is T, it follows that $(f_\pi(v_{j_1}), \ldots, f_\pi(v_{j_{m_i}})) \in \rho_i$ so that $f_\pi \in \mathcal{S}(P)$. Clearly $f_\pi \leq_T \pi$. Thus each infinite path π through T gives rise to a solution of (V, A, C). Vice versa, if f is a solution of (V, A, C), then for each i it must be the case that $f(i) \in dom(\rho_{G(i)})$ so that we can effectively find the unique t_i such that $f(i) \in W_{H(i),t_i} - W_{H(i),t_i-1}$ from $f(i)$. It then follows that $\pi_f = (t_0, f(0), t_1, f(1), t_2, f(2), \ldots)$ is an infinite path through T and $\pi_f \leq_T f$. Moreover, it is easy to see that $f_{\pi_f} = f$. Thus there is an effective one-to-one degree preserving correspondence between $\mathcal{S}(P)$ and $[T]$.

Now if $P = (V, A, C)$ is bounded, then we claim that T is bounded. That is, if $(\eta_1, \ldots, \eta_k) \in T$, then it must be the case that if $2n \leq k$, then $\eta_{2n} \in dom(\rho_{G(n-1)})$ so that there are only finitely many possible values for η_{2n}. Similarly, if $2n+1 \leq k$, then $W_{H(n),\eta_{2n+1}} - W_{H(n),\eta_{2n+1}-1} \neq \emptyset$ so that again there are only finitely possible values of η_{2n+1}. Thus T is bounded if P is bounded. Moreover, if P is recursively bounded, then it is easy to see that we can effectively find the possible values of η_{2n} and η_{2n+1} in each case so that T will be highly recursive in that case. $\square$

Next we want to reverse Theorem 2.6. That is, we want to show that given an recursive tree T, we can construct recursive constraint satisfaction problem P such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P)$.

Theorem 2.7. *If T is a recursive tree contained in $\omega^{<\omega}$, then there is an infinite recursive constraint satisfaction problem $P = (V, A, C)$ such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P)$. Moreover, if T is finitely branching, then P is bounded and, if T is highly recursive, then P is recursively bounded.*

Proof: Given T, construct a new recursive tree T^* by putting the empty sequence into T^*, the sequence of length i $(i, i, \ldots, i)$ in T^* for all $i \geq 1$, and putting in $(0, \eta_1, \ldots, \eta_n)$ into to T^* for all $(\eta_1, \ldots, \eta_n) \in T$. That is, T^* is constructed from T by adding a copy of T above the node (0) and adding nodes with constant strings of length i, $(i, \ldots, i)$ in T^* for all $i \geq 1$. Clearly there is an effective one-to-one degree preserving correspondence between $[T]$ and $[T^*]$.

Let $C = \{C_i : i \in \omega\}$ where $C_i = \langle s_i, \rho_i \rangle$ and $s_i = (v_0, \ldots, v_i)$ and ρ_i be the $i+1$-relation such that ρ_i contains $\{(\eta_0, \ldots, \eta_i) \in T^* \ \& \ \eta_0 = 0\}$ plus constant string of length $i+1$, $(i+1, \ldots, i+1)$

Now if $\pi = (\pi_0, \pi_1, \ldots)$ is in $[T^*]$, then it must be the case that $\pi_0 = 0$, Hence if $f_\pi(v_j) = \pi_j$ for $j \geq 0$, then for each i, $(f_\pi(v_0), \ldots, f_\pi(v_i)) \in \rho_i$ since $(\pi_0, \ldots, \pi_i) \in T^*$. Thus $f_\pi \in \mathcal{S}(P)$. Vice versa, if $f \in \mathcal{S}(P)$, then consider the path $\pi_f = (f(v_0), f(v_1), \ldots)$. Then since $(f(v_0), f(v_1), \ldots, f(v_i)) \in \rho_i$, it must be the case that $(f(v_0), f(v_1), \ldots, f(v_i)) \in T^*$. Hence $\pi_f \in [T]$. Moreover it is easy to see that $\pi_{f_\pi} = \pi$ so that there is an effective one-to-one correspondence between $\mathcal{S}(P)$ and $[T]$.

Note that if T is bounded, then there are only finitely many nodes of length n in

T^* which extend (0) for each n and, hence, it easily follows that each ρ_i is finite. Similarly, if T is recursively bounded, then we can effectively find the set of nodes of length n for each n from which it follows that we can compute a $g(n)$ such that all nodes of length n in T^* lie in $\{0, \ldots, g(n)\}^n$. It then easily follows that P is recursively bounded. $\square$

We note that in the construction of Theorem 2.7, we used n-relations for every $n \geq 1$ in the infinite recursive CSP problem P. It is natural to ask whether this is necessary. Our next result will show that this not case. Namely, we could have used only unary and binary relations.

Theorem 2.8. *If T is a recursive tree contained in $\omega^{<\omega}$, then there is an infinite recursive constraint satisfaction problem $P^* = (V^*, A^*, C^*)$ such that there is an effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P)$ and all relations in the constraints of P are either unary relations or binary relations. Moreover, if T is finitely branching, then P is bounded and, if T is highly recursive, then P is recursively bounded.*

Proof: Let T^* be as in Theorem 2.7. Then we consider the following infinite recursive constraint satisfaction problem $P^* = (V^*, A^*, C^*)$. First we let $V^* = \{v_i : i \in \omega\}$ and $A^* = \omega$. Then we will have two types of constraints. For each $n \geq 0$, we let $C_{2n} = \langle s_{2n}, \rho_{2n} \rangle$ where $s_{2n} = (v_n)$ and $\rho_{2n} = \{c((\eta_1, \ldots, \eta_{n+1})) : (\eta_1, \ldots, \eta_{n+1}) \in T^*\}$. That is, ρ_{2n} is the set of codes of nodes of length $n+1$ in T^*. Then we let $C_{2n+1} = \langle s_{2n+1}, \rho_{2n+1} \rangle$ where $s_{2n+1} = (v_n, v_{n+1})$ and

$$\rho_{2n+1} = \{[c((\eta_1, \ldots, \eta_{n+1})), c((\eta_1, \ldots, \eta_{n+2})] : (\eta_1, \ldots, \eta_{n+2}) \in T^*\}.$$

It is easy to see that for any solution f^* of P^*, it must be the case $f^*(n)$ is the code of a node $(\eta_1^*, \ldots, \eta_{n+1}^*)$ in T^*. However, the fact that all the constraints of the form C_{2n+1} are satisfied implies that $f^*(0) \sqsubseteq f^*(1) \sqsubseteq \cdots$ so that f must code an infinite path through T^*. Thus it is easy see that there is an effective one-to-one degree preserving correspondence between $\mathcal{S}(P^*)$ and T^*. It also easy to see that if T is bounded, then P^* is bounded and if T is highly recursive, then P^* is recursively bounded. $\square$

3 Index Set Results

In this section, we will exploit the uniformities in the proofs of Theorems 2.6 and 2.7 to develop some index set results. To this end, we shall consider general recursive constraint satisfaction problems (V, A, C) where

1. V is a recursive subset of $\{v_i : i \in \omega\}$,

2. A is a recursive subset of ω,
3. for each $C_i = \langle s_i, \rho_i \rangle \in C$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$, ρ_i is recursive relation, and
4. the set of codes of constraints in C is a recursive set.

Recall that if $C_i = \langle s_i, \rho_i \rangle$ where $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$, then the code of C_i is a triple $code(C_i) = [m_i, a_i, b_i]$ where a_i is a code of the m_i-tuple $(j_1, \ldots, j_{m_i})$ and b_i is a recursive index of ρ_i. Then we will say that x is a code of a recursive constraint satisfaction problem (V, A, C) if $x = [e, f, g]$ where e is a recursive index of the set $\{i : v_i \in V\}$, f is a recursive index of A, and g is a recursive index of $\{code(C_i) : C_i \in C\}$.

There are a number of reasons why a code $[e, f, g]$ can fail to be a code of an infinite recursive constraint satisfaction problem. We list them as (**A**) – (**E**) below.

(A) ϕ_e is not a total recursive function or the range of ϕ_e is not contained in $\{0, 1\}$.

(B) ϕ_f is not a total recursive function or the range of ϕ_f is not contained in $\{0, 1\}$.

(C) ϕ_g is not a total recursive function or the range of ϕ_g is not contained in $\{0, 1\}$.

(D) $\phi_g([m_i, a_i, b_i]) = 1$, but either a_i is not the code of m_i-tuple, ϕ_{b_i} is not total or the range of ϕ_{b_i} is not contained in $\{0, 1\}$, or there is a z such that $\phi_{b_i}(z) = 1$, but z is not the code of m_i-tuple from A^{m_i}.

(E) There is a variable $v_i \in V$ such there there is no constraint $C_j = \langle s_j, \rho_j \rangle \in C$ such that v_i occurs in C.

When we say that we have checked that $[e, f, g]$ is a code of a recursive constraint satisfaction problem up to stage n, we mean that

(i) we have computed $\phi_e(0), \phi_f(0), \phi_g(0), \phi_e(1), \phi_f(1), \phi_g(2), \ldots, \phi_e(n), \phi_f(n), \phi_g(n)$ and all these values are in $\{0, 1\}$,

(ii) for each $y \leq n$ such that $\phi_{b_i}(y) = 1$ and $y = [m_i, a_i, b_i]$, a_i is the code of an m_i-tuple from V and we have computed $\phi_{b_i}(0), \ldots, \phi_{b_i}(n)$, all these values are in $\{0, 1\}$, and for each $j \leq n$, if $\phi_{b_i}(j) = 1$, then j is the code of m_i-tuple of elements from A, and

(iii) for each $i \leq n$ such that $\phi_e(i) = 1$, we have computed $\phi_g(0), \phi_g(1), \ldots$ until we have found the least p such that $\phi_g(p) = 1$, $p = [m_r, a_r, b_r]$, a_r is the code of an m_r-tuple from V which contains v_i and we have computed $\phi_{b_r}(0), \ldots, \phi_{b_r}(p)$, all these values are in $\{0, 1\}$, and for each $j \leq p$, if $\phi_{b_r}(j) = 1$, then j is the code of m_r-tuple of elements from A.

Of course, it may be that some of these computations do not converge. For example, if $\phi_e(1)$ is undefined, then we can never check $[e, f, g]$ is a code of a recursive constraint satisfaction problem up to stage 1. In fact, it is easy to see that if any of $(\mathbf{A}) - (\mathbf{B})$ above hold, then there will be some n such that we can not check that $[e, f, g]$ is a code of a recursive constraint satisfaction problem up to stage n. However, if $[e, f, g]$ is a code of a recursive constraint satisfaction problem, then we will be able to check that $[e, f, g]$ is a code of a recursive constraint satisfaction problem for all stages n.

Recall that x is a code of a recursive tree $T \subseteq \omega^{<\omega}$, if x is the recursive index of the set of codes of the nodes of T. Again, there are several reasons why x could fail to be the code of a recursive tree T.

(I) ϕ_x is not total or the range of ϕ_x is not contained in $\{0, 1\}$.

(II) ϕ_x is total, and the range of ϕ_x is contained in $\{0, 1\}$, but T is not closed under initial segments. That is, there is are nodes $\alpha \sqsubset \beta$ such that if the code of α equals a and the code of β equals b, then $\phi_x(a) = 0$ and $\phi_x(b) = 1$.

Now let $a_0 < a_1 < \cdots$ be an effective list of all the codes of nodes in $\omega^{<\omega}$. Then we say that we have *checked* x is a code of recursive tree up to stage n, if
(i) we have computed $\phi_x(i)$ for all $i \leq a_n$ and $\phi_x(i) = 0$ for all $i \in \{0, \ldots, a_n\} \setminus \{a_0, a_1, \ldots, a_n\}$ and
(ii) if $i \in \{a_0, a_1, \ldots, a_n\}$, $\phi_x(a_i) = 1$, and a_i is the code of node β, then, for all $\alpha \sqsubset \beta$, the value of ϕ_x on the code of α is also 1.

This given, we are now in a position to state the uniform versions of Theorems 2.6 and 2.7. That is, we have the following theorem .

Theorem 3.1. *1. There is a recursive one-to-one function q such that for all x,*
(a) if x is the code of a recursive constraint satisfaction problem $P = (V, A, C)$, then $q(x)$ is the code of a recursive tree T such that there is an effective degree preserving one-to-one correspondence between $\mathcal{S}(P)$ and $[T]$ and
(b) if x is not the code of a recursive constraint satisfaction problem, then $q(x)$ is not the code of a recursive tree contained in $\omega^{<\omega}$.

2. There is a recursive one-to-one function p such that for all x,
(a) if x is the code of a recursive tree T, then $p(x)$ is the code of a recursive constraint satisfaction problem $P = (V, A, C)$ such that there is an effective degree preserving one-to-one correspondence between $\mathcal{S}(P)$ and $[T]$ and
(b) if x is not the code of a recursive tree contained in $\omega^{<\omega}$, then $p(x)$ is not of a recursive constraint satisfaction problem.

Proof: To prove (1), we will use the proof of Theorem 2.6. In Theorem 2.6, we gave a construction of the desired tree T given that we started with an infinite recursive constraint satisfaction problem $P = (V, A, C)$ where $V = \{v_i : i \in \omega\}$ and

$C = \{C_i : i \in \omega\}$. However, even if $x = [e, f, g]$ is the code of recursive constraint satisfaction problem $P = (V, A, C)$, it is not guaranteed that either V or C is infinite. To this end, we will construct a new recursive constraint satisfaction problem $P^* = (V^*, A^*, C^*)$ where V^* and C^* are infinite and recursive and there is an effective degree preserving one-to-one correspondence between $\mathcal{S}(P)$ and $\mathcal{S}(P^*)$. First let $2V = \{v_{2i} : v_i \in V\}$, $2A = \{2i : i \in A\}$, and $2C = \{2C_i : C_i \in C\}$. Here for any constraint $C_i = \langle s_i, \rho_i \rangle$ in C such $s_i = (v_{j_1}, \ldots, v_{j_{m_i}})$ and $\rho_i \subseteq A^{m_i}$, $2C_i = \langle 2s_i, 2\rho_i \rangle$ where $2s_i = (v_{2j_1}, \ldots, v_{2j_{m_i}})$ and $2\rho_i = \{(2a_1, \ldots 2a_{m_i}) : (a_1, \ldots, a_{m_i}) \in \rho_i\}$. Then we let $P* = (V^*, A^*, C^*)$ where
(i) $V^* = 2V \cup \{v_{2i+1} :\in \omega\}$,
(ii)$A^* = 2A \cup \{1\}$, and
(iii) $C^* = 2C \cup \{D_{2i+1} : i \in \omega\}$ where $D_{2i+1} = \langle (v_{2i+1}), \{1\} \rangle$. That is, each D_{2i+1} has single variable v_{2i+1} and the corresponding constraint relation is unary relation consisting the set $\{1\}$.

It is easy to see that given a solution ϕ of (V, A, C), the function ϕ^* which maps variable v_{2j} to $\phi(v_i)$ and maps all the variables v_{2i+1} to 1 is a solution to P^* and, moreover, it the case that any solution to P^* is of the form ϕ^* for some solution to ϕ of P. Thus there is an effective degree preserving one-to-one correspondence between $\mathcal{S}(P)$ and $\mathcal{S}(P^*)$. Since the construction of P^* from P is uniform, it easily follows that there is a recursive one-to-one function h such that $h(x)$ is the code of P^* if x is the code of P and $h(x)$ is not the code of recursive constraint satisfaction problem if x is not the code of recursive constraint satisfaction problem.

We can now apply the construction of Theorem 2.6 to P^* if we interpret the sequence of variables $v_0, v_1, \ldots$ as an effective list of all variables that occur V^* and we interpret the sequence of constraints $C_0, C_1, \ldots$ as an effective sequence of all constraints of C^*. We also modify the construction of the tree T in this case so we do not put a node of length n into T unless we have checked up to stage n that $h(x)$ is the code of a recursive constraint satisfaction problem. This way, if P^* is not a recursive constraint satisfaction problem, then our construction will not give a total decision procedure to decide which nodes are in T. It is then easy to see that our construction is uniform so that the desired function q is nothing but the code of the set of instructions to compute P^* and then T from x. This proves (1).

The proof of (2) is similar. That is, Theorem 2.7 gives a uniform construction given a code x of a recursive tree T to produce a desired recursive constraint problem P. The only thing that we have to modify in this construction is to insist that we do not define C_n until we have checked that x is the code of tree T up to stage n. In this way, if x is not the code of a recursive tree, then the characteristic function of the set of codes in C will not be defined and so P will not be a recursive constraint satisfaction problem. It is then easy to see that our construction is uniform so that the desired function p is nothing but the code of the set of instructions to compute

P from x. $\square$

Let us recall that a subset A of ω is said to be Σ_n^m-complete (respectively, Π_n^m-complete) if A is Σ_n^m (respectively, Π_n^m) and any Σ_n^m (respectively, Π_n^m) set B is many-one reducible to A. Here we say B is many-one reducible to A if there is a recursive function f such that, for any b, $b \in B$ if and only if $f(b) \in A$. A subset A of ω is said to be D_n^m if it is the difference of two Σ_n^m sets. A subset A of ω is said to be D_n^m complete if it is D_n^m and any D_n^m set B is many-one reducible to A. Cenzer and Remmel [CR98a, CR98b] proved a large number of results on the complexity of index sets based on primitive recursive indices for trees. That is, suppose that $\pi_0, \pi_1, \ldots$ is an effective enumeration of the primitive recursive functions from ω to $\{0, 1\}$. Let

$$U_e = \{\emptyset\} \cup \{\sigma : (\forall \tau \preceq \sigma)\pi_e(\langle\tau\rangle) = 1\}.$$

It is clear that each U_e is a primitive recursive tree. Observe also that if $\{\sigma : \pi(\sigma) = 1\}$ is a primitive recursive tree, then U_e will be that tree. Thus every primitive recursive tree occurs in our enumeration U_e. Now it is well known that for any recursive tree T, there is a primitive recursive tree $T*$ such that $[T] = [T^*]$ and, moreover, there is recursive function g such that if x is a recursive index for T, then $g(x)$ is such that $g(x)$-th primitive recursive function computes the characteristic function of T^*. For any property Q of trees, Cenzer and Remmel [CR98a] considered the index set $I_P(Q)$ which is equal to set of all e such U_e has property Q. In our case, need to consider index sets based on recursive indices. That is, we will want to consider the index set $I_R(Q)$ equal to the set of all e such that e is a recursive index of a tree T_e and T_e has property Q. It is easy to see by writing out the definition that the property of e being the recursive index of a recursive tree is a Π_2^0 property. The proofs of the level of complexity that Cenzer and Remmel proved for index sets $I_P(Q)$ in the arithmetic hierarchy also hold for the index sets $I_R(P)$ as long as the complexity is above Π_2^0. Thus for example, the following results follow from the results of Cenzer and Remmel in [CR98a, CR98b].

Theorem 3.2.

1. *Let $Q_1(e)$ be the property: 'e is a recursive index of a recursive finitely branching tree'. Then $I_R(Q_1)$ is Π_3^0 complete.*

2. *Let $Q_2(e)$ be the property: 'e is a recursive index of a highly recursive tree'. Then $I_R(Q_2)$ is Σ_3^0 complete.*

3. *Let $Q_3(e)$ be the property: 'e is a recursive index of a recursive finitely branching tree and $[T_e]$ is nonempty'. Then $I_R(Q_3)$ is Π_3^0 complete.*

4. *Let $Q_4(e)$ be the property: 'e is a recursive index of a highly recursive tree and $[T_e]$ is nonempty'. Then $I_R(Q_4)$ is Σ_3^0 complete.*

5. *Let $Q_5(e)$ be the property: 'e is a recursive index of a recursive tree and $[T_e]$ is nonempty'. Then $I_R(Q_5)$ is Σ_1^1 complete.*

6. *Let $Q_6(e)$ be the property: 'e is a recursive index of a recursive tree and $[T_e]$ is empty'. Then $I_R(Q_6)$ is Π^1_1 complete.*

One can use these results to immediately prove index set results for recursive constraint satisfaction problems. That is, for any property Q of recursive CSPs, let $I_C(Q)$ denote the set of e such e is a recursive index of a recursive CSP problem P_e and P_e has property Q. Then we can use Theorem 3.1 to prove analogues of the index set results in Theorem 3.2 for the corresponding sets $I_C(Q_i)$ for $i = 1, \ldots 6$. That is, we have the following results.

Theorem 3.3. 1. *Let $Q_1(e)$ be the property: 'e is a recursive index of a bounded infinite recursive CSP problem'. Then $I_C(Q_1)$ is Π^0_3 complete.*

2. *Let $Q_2(e)$ be property: 'e is a recursive index of a recursively bounded infinite recursive CSP problem'. Then $I_C(Q_2)$ is Σ^0_3 complete.*

3. *Let $Q_3(e)$ be the property: 'e is a recursive index of a bounded infinite recursive CSP problem and $\mathcal{S}(P_e)$ is nonempty'. Then $I_C(Q_3)$ is Π^0_3 complete.*

4. *Let $Q_4(e)$ be the property: 'e is a recursive index of recursively bounded infinite recursive CSP problem and $\mathcal{S}(P_e)$ is nonempty'. Then $I_C(Q_4)$ is Σ^0_3 complete.*

5. *Let $Q_5(e)$ be the property: 'e is a recursive index of an infinite recursive CSP problem and $\mathcal{S}(P_e)$ is nonempty'. Then $I_C(Q_5)$ is Σ^1_1 complete.*

6. *Let $Q_6(e)$ be the property: 'e is a recursive index of an infinite recursive CSP problem and $\mathcal{S}(P_e)$ is empty'. Then $I_C(Q_6)$ is Π^1_1 complete.*

Proof: All of these results can be proved in the same way. First, one establishes the upper bound in each case by simply writing our the formal definition and checking it is has the appropriate form. Then to establish the completeness in each case, one uses Theorem 3.1 and the corresponding completeness result from Theorem 3.2. □

Cenzer and Remmel [CR98a, CR98b] also proved a large number of index set results about the cardinality of the set of infinite paths. For example, they proved the following results.

Theorem 3.4. 1. *Let c be a positive integer c and let $Q_1^{=c}(e)$be the following property: 'e is a recursive index of a recursive finitely branching tree and $|[T_e]| = c$'. Then $I_R(Q_1^{=c})$ is Π^0_3 complete if $c = 1$ and is D^0_3 complete if $c > 1$.*

2. *Let c be a positive integer and $Q_2^{=c}(e)$ be the property: 'e is a recursive index of a highly recursive tree and $|[T_e]| = c$'. Then $I_R(Q_2^{=c})$ is Σ^0_3 complete.*

3. *Let c be a positive integer and $Q_3^{=c}(e)$ be the property: 'e is a recursive index of a recursive tree and $|[T_e]| = c$'. Then $I_R(Q_3^{=c})$ is Π^1_1 complete.*

4. *Let $Q_1^{fin}(e)$ be the property: 'e is a recursive index of a recursive finitely branching tree and $|[T_e]|$ is finite'. Then $I_R(Q_1^{fin})$ is Σ_4^0 complete.*

5. *Let $Q_2^{fin}(e)$ be the property: 'e is a recursive index of a highly recursive tree and $|[T_e]|$ is finite'. Then $I_R(Q_2^{fin})$ is Σ_3^0 complete.*

6. *Let $Q_3^{fin}(e)$ be the property: 'e is a recursive index of a recursive tree and $|[T_e]|$ is finite'. Then $I_R(Q_3^{fin})$ is Π_1^1 complete.*

7. *Let $Q_1^{infin}(e)$ be the property: 'e is a recursive index of a recursive finitely branching and $|[T_e]|$ is infinite'. Then $I_R(Q_1^{infin})$ is Π_4^0 complete.*

8. *Let $Q_2^{infin}(e)$ be the property: 'e is a recursive index of a highly recursive tree and $|[T_e]|$ is infinite'. Then $I_R(Q_2^{infin})$ is D_3^0 complete.*

9. *Let $Q_3^{infin}(e)$ be the property: 'e is a recursive index of a recursive tree and $|[T_e]|$ is infinite'. Then $I_R(Q_3^{infin}$ is Σ_1^1 complete.*

Once again all these results can be transfered to index set results for recursive CSPs.

Theorem 3.5. 1. *Let c be a positive integer and let $Q_1^{=c}(e)$ be the property: 'e is a recursive index of a bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)| = c$'. Then $I_C(Q_1^{=c})$ is Π_3^0 complete if $c = 1$ and is D_3^0 complete if $c > 1$.*

2. *Let c be a positive integer c and let $Q_2^{=c}(e)$ be the property: 'e is a recursive index of a recursively bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)| = c$'. Then $I_C(Q_2^{=c})$ is Σ_3^0 complete.*

3. *Let c be a positive integer c and let $Q_3 = c(e)$ be the property: 'e is a recursive index of an infinite recursive CSP problem and $|\mathcal{S}(P_e)| = c$'. Then $I_C(Q_3^{=c})$ is Π_1^1 complete.*

4. *Let $Q_1^{fin}(e)$ be the property: 'e is a recursive index of a bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is finite'. Then $I_C(Q_1^{fin})$ is Σ_4^0 complete.*

5. *Let $Q_2^{fin}(e)$ be the property: 'e is a recursive index of a recursively bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is finite'. Then $I_C(Q_2^{fin})$ is Σ_3^0 complete.*

6. *Let $Q_3^{fin}(e)$ be the property: 'e is a recursive index of an infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is finite'. Then $I_C(Q_3^{fin})$ is Π_1^1 complete.*

7. *Let $Q_1^{infin}(e)$ be the property: 'e is a recursive index of a bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is infinite'. Then $I_C(Q_1^{infin})$ is Π_4^0 complete.*

8. *Let $Q_2^{infin}(e)$ be the property: 'e is a recursive index of a recursively bounded infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is infinite'. Then $I_C(Q_2^{infin})$ is D_3^0 complete.*

9. *Let $Q_3^{infin}(e)$ be the property: 'e is a recursive index of an infinite recursive CSP problem and $|\mathcal{S}(P_e)|$ is infinite'. Then $I_C(Q_3^{infin})$ is Σ_1^1 complete.*

Similarly Cenzer and Remmel [CR98a] proved a large number of index set results for the number of infinite recursive paths through recursive trees and all of those results can be transfered to index sets for the number of recursive solutions to recursive CSP problems. For example, Cenzer and Remmel [CR98a, CR98b] proved the following.

Theorem 3.6. 1. *Let $Q_1^{\exists rec}(e)$ be the property: 'e is a recursive index of a recursive finitely branching tree and there is a recursive path through T_e'. Then $I_R(Q_1^{\exists rec})$ is D_3^0 complete.*

2. *Let $Q_2^{\exists rec}(e)$ be the property: 'e is a recursive index of a highly recursive tree and there is a recursive path through T_e'. Then $I_R(Q_2^{\exists rec})$ is Σ_3^0 complete.*

3. *Let $Q_3^{\exists rec}(e)$ be the property: 'e is a recursive index of a recursive tree and there is a recursive path through T_e'. Then $I_R(Q_3^{\exists rec})$ is Σ_3^0 complete.*

4. *Let $Q_4^{norec}(e)$ be the property: 'e is a recursive index of a recursive finitely branching tree, $[T_e]$ is nonempty, and there is no recursive path through T_e'. Then $I_R(Q_4^{norec})$ is Π_3^0 complete.*

5. *Let $Q_5^{norec}(e)$be the property: 'e is a recursive index of a highly recursive tree, $[T_e]$ is nonempty, and there is no recursive path through T_e'. Then $I_R(Q_5^{norec})$ is D_3^0 complete.*

6. *Let $Q_6^{norec}(e)$ be the property: 'e is a recursive index of a recursive tree, $[T_e]$ is nonempty, and there is no recursive path through T_e'. Then $I_R(Q_6^{norec})$ is Σ_1^1 complete.*

Again we can transfer these results to get similar result for the corresponding index sets for recursive solutions of recursive CSP problems.

4 The degrees of solutions to recursive CSP problems

One can also use Theorem 3.1 to transfer a large number or results concerning the degrees of infinite paths through recursive trees to results about the degrees of solutions to CSP problems. In this section, we shall give a sample of both positive and negative results of this kind. References for all the results on recursive tree that lie behind the results stated in this section can be found in the forthcoming book by Cenzer and Remmel [CRta].

4.1 Positive results for recursive constraint problems

The results of sections 2 and 3 show that whenever we have a recursive constraint satisfaction problem with the unique solution, we can produce a recursive tree with a unique infinite path such that the Turing degrees of the solution and the branch are the same. Conversely, given a tree with a unique infinite path, we can produce a recursive constraint satisfaction problem with a unique solution such that the Turing degrees of the infinite path and of the solution are the same.

The degrees of elements of Π^0_1-classes have been extensively studied in recursion theory. It follows from Theorems 2.6, 2.7, and 3.1 that we can immediately transfer results about degrees of elements of Π^0_1-classes to results about the degrees of solutions of recursive constraint satisfaction problems. First we give a sample of so-called basis theorems. That is, we state several theorems which state that one can always find solutions in a certain class.

Corollary 4.1 (Positive results for recursive infinite constraint satisfaction problems). *Suppose P is a recursive infinite constraint satisfaction problem with a solution. Then the following hold.*

1. *P has a solution which is recursive in a complete Σ^1_1 set.*
2. *If P has only finitely many solution, then each solution is hyperarithmetic.*
3. *If P has countably many solutions, then P has a solution which is hyperarithmetic.*

Corollary 4.2 (Positive results for recursively bounded recursive infinite constraint satisfaction problems). *Suppose that P is a recursively bounded recursive infinite constraint satisfaction problem with a solution. Then the following hold.*

1. *P has a solution whose Turing jump is recursive in $\mathbf{0}'$.*
2. *If P has only finitely many solutions, then each of these solutions is recursive.*
3. *If P has countably many solutions, then P has a recursive solution.*
4. *There is a solution f of P in an r.e. degree.*
5. *There exist solutions f_1 and f_2 of P such that any function, recursive in both f_1 and f_2, is recursive.*
6. *If P has no recursive solution, then there is a nonzero r.e. degree $\mathbf{a}$ such that P has no solution recursive in $\mathbf{a}$.*

The next set of corollaries follow because a recursive finitely branching tree is automatically highly recursive in $\mathbf{0}'$.

Corollary 4.3 (Positive results for bounded recursive constraint satisfaction problems). *Suppose P is bounded recursive infinite constraint satisfaction problem which has a solution. Then the following hold.*

1. *There is a solution f of P whose Turing jump is recursive in $\mathbf{0}''$, the Turing jump of $\mathbf{0}'$.*
2. *If P has only finitely many solutions, then each of these solutions is recursive in $\mathbf{0}'$.*
3. *If P has countably many solution, then P has a solution which is recursive in $\mathbf{0}'$.*
4. *There is a solution f which is in some r.e. degree in $\mathbf{0}'$.*
5. *There are solutions f_1 and f_2 such that any function, recursive in both f_1 and f_2, is recursive in $\mathbf{0}'$.*
6. *If P has no solution which is recursive in $\mathbf{0}'$, then there is a nonzero degree $a >_T \mathbf{0}'$ such that a is r.e. $\mathbf{0}'$ and such that P has no solutions recursive in a.*

4.2 Negative results for recursive infinite constraint problems

Next we state a selection of results that can be considered negative results in that they show that there are recursive infinite constraint problems whose solution set is very restricted.

Corollary 4.4 (Negative results for recursive infinite constraint problems).

1. *There exists a infinite recursive constraint satisfaction problem P such that P has a solution but P has no solution which is hyperarithmetic.*
2. *For any recursive ordinal α, there is a recursive infinite constraint satisfaction problem P such that P has a unique solution f and $f \equiv_T 0^{(\alpha)}$.*

Using well-known recursion-theoretic facts about recursively bounded Π^0_1 classes we get:

Corollary 4.5 (Negative results for recursively bounded recursive infinite constraint satisfaction problems).

1. *There exists a recursively bounded recursive infinite constraint satisfaction problem P_1 such that P_1 has no recursive solutions (although P_1 possesses $2^{\aleph_0}$ solutions).*

2. *There exists a recursively bounded recursive infinite constraint satisfaction problem P_2 such that P_2 possesses $2^{\aleph_0}$ solutions and any two solutions $f_1 \neq f_2$ of P_2 are Turing incomparable.*

3. *If $\mathbf{a}$ is a Turing degree and $\mathbf{0} <_T \mathbf{a} <_T \mathbf{0}'$, then there exists a recursively bounded recursive infinite constraint satisfaction problem P_3 such that P_3 has $2^{\aleph_0}$ solutions, a solution of degree $\mathbf{a}$ but P_3 has no recursive solution.*

4. *There exists a recursively bounded recursive infinite constraint satisfaction problem P_4 such that if $\mathbf{a}$ is the degree of any solution of P_4 and $\mathbf{b}$ is a r.e. degree with $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T \mathbf{0}'$.*

5. *If $\mathbf{c}$ is any r.e. degree, then there exists a recursively bounded recursive infinite constraint satisfaction problem P_5 such that the set of r.e. degrees which contain solutions of P_5 equals the sets of r.e. degrees $\geq_T \mathbf{c}$.*

6. *There exists a recursively bounded recursive infinite constraint satisfaction problem P_6 such that if f is solution for P_6 where $f <_T \mathbf{0}'$, then there exists a nonrecursive r.e. set A such $A <_T f$.*

We can relativize all the results in Corollary 4.2 to an $\mathbf{0}'$ oracle for bounded recursive constraint satisfaction problems This is due to the following result of Jockusch, Lewis, and Remmel.

Theorem 4.6 ([JLR91]). *For any tree T which is highly recursive in $\mathbf{0}'$, there is a recursive finitely branching tree $S \subseteq \omega^{<\omega}$ with an effective one-to-one degree preserving correspondence between $[T]$ and $[S]$.*

Encoding highly recursive in $\mathbf{0}'$ trees by binary trees gives us now results on bounded recursive constraint satisfaction problems.

Corollary 4.7 (Negative results for bounded recursive infinite constraint satisfaction problems).

1. *There exists a bounded recursive infinite constraint satisfaction problem P_1 such that P_1 has no solution which is recursive in $\mathbf{0}'$, although P possesses $2^{\aleph_0}$ solutions.*

2. *There exists a bounded recursive infinite constraint satisfaction problem P_2 such that P_2 possesses $2^{\aleph_0}$ solutions and any two solutions $f_1 \neq f_2$ of P_2 have the property that $f_1 \bigoplus \mathbf{0}' \not\equiv_T f_2 \bigoplus \mathbf{0}'$.*

3. *If $\mathbf{a}$ is a Turing degree and $\mathbf{0}' <_T \mathbf{a} <_T \mathbf{0}''$, then there exists a bounded recursive infinite constraint satisfaction problem P_3 such that P_3 has $2^{\aleph_0}$ solutions, a solution of degree $\mathbf{a}$ but P_3 has no solution which is recursive in $\mathbf{0}'$.*

4. *There exists a bounded recursive infinite constraint satisfaction problem P_4 such that P has $2^{\aleph_0}$ solutions, and if $\mathbf{a}$ is the degree of any solution of P_4 and $\mathbf{b}$ is a degree which is r.e. in $\mathbf{0}'$ with $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T \mathbf{0}''$.*

5. *If $\mathbf{c} \geq_T \mathbf{0}'$ is any degree which is r.e. in $\mathbf{0}'$, then there exists a bounded recursive infinite constraint satisfaction problems P_5 such that the set of degrees which are r.e. in $\mathbf{0}'$ and which contain solutions of P_5 equals the sets of degrees $\geq_T \mathbf{c}$ which are r.e. in $\mathbf{0}'$.*

6. *There exists a bounded recursive infinite constraint satisfaction problems P_6 such that if f is solution for P_6 where $\mathbf{0}' \leq_T f <_T \mathbf{0}''$, then there exists a set A such $\mathbf{0}' <_T A <_T f$ and A is r.e. in $\mathbf{0}'$.*

5 Conclusions and further research

As happens often in recursive mathematics, the investigations from mathematics or computer science inspire research on problems that generalize finite problems to the recursive case. The research reported in this paper is of this kind, and is directly inspired by the recent progress on constraint satisfaction. It is quite clear that even more restrictive classes of constraint satisfaction problems can be considered, for instance we could restrict ourselves to the case where the set of variables V, the underlying domain A, and all the relations that appear in the constraints are recognized by finite automaton. Such CSP can be considered automatic structures in the sense of [KN95] and [KNRS07]. Such classes may be related to unbounded model checking and other tasks grounded in electronic design automation practice. Similarly, one could consider the class of infinite CSP's where the set of variables V, the underlying domain A, and all the relations that appear in the constraints are in some complexity class such a polynomial time, polynomial space, etc.. It is then a natural questions to ask for conditions on infinite CSP's that ensure that there is a solution of which is automatic, polynomial time, recursive, etc..

Yet another puzzling question related to recursive satisfaction problem is a possibility of generalization of *dichotomy theorems* that lie on the crossroads of computational universal algebra and computer science. The fundamental theorem of Schaefer [Sch78] classifying (finite) Boolean constraint satisfaction problems according to the description of these constraints (but on a deeper level according to their invariants) and its generalization by Bulatov [Bul02] to the case of 3-element domains raise a possibility of the presence of dichotomies in the recursive case, as well. We hope that if such generalizations are, indeed, possible, our paper is the first step in this direction.

Acknowledgements

This research of the first author was supported by the National Science Foundation under Grant IIS-0325063. This research of the second author was supported by the National Science Foundation under Grant DMS 0654060.

References

[Apt03] Apt, K.R., *Principles of Constraint Programming*, Cambridge University Press, 2003.

[Bul02] Bulatov, A.A. A Dichotomy Theorem for Constraints on Three-Element Set. *Proceedings of FOCS 2002*, pages 649–658, 2002.

[CR98a] Cenzer, D. and Remmel, J.B. Π^0_1 Classes in Mathematics, *Handbook of Recursive Mathematics, Vol 2.*, Studies in Logic and the Foundations of Mathematics, Vol. 139, (eds. Y. Ershov, S. Goncharov, A. Nerode, and J.B. Remmel), pages 632–822, Elsevier, 1998.

[CR98b] Cenzer, D. and Remmel, J.B. Index sets for Π^0_1 classes, *Annals of Pure and Applied Logic* 93:3–61, 1998.

[CRta] Cenzer, D. and Remmel, J.B. Effectively Closed Sets, ASL Lecture Notes in Logic, to appear. Preliminary version available at `http://www.math.ufl.edu/~cenzer/`.

[CCT87] Cook, W.J., Coullard, C. and Turan, G. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics* 18:25-38, 1987.

[Dec03] Dechter, R. *Constraint Processing*, Morgan Kaufmann, 2003.

[Hin78] Hinman, P. *Recursion-theoretic Hierarchies*, Springer-Verlag, 1978.

[Hoo00] Hooker, J.N *Logic-Based Methods for Optimization*, Wiley, 2000.

[JLR91] Jockusch, C.G., Lewis, A. and Remmel, J.B. Π^0_1 Classes and Rado's Selection Principle. *Journal of Symbolic Logic*, 56:684–693, 1991.

[JS72a] Jockusch, C.G. and Soare, R.I. Π^0_1 Classes and Degrees of Theories. *Transactions of American Mathematical Society*, 173:33–56, 1972.

[JS72b] Jockusch, C.G. and Soare, R.I. Degrees of members of Π^0_1 classes. *Pacific Journal of Mathematics*, 40:605-616, 1972.

[KN95] Khoussainov, B. and Nerode, A. Automatic Presentations of Structures. In: *Logical and Computational Complexity*, Springer Lecture Notes in Computer Science 960, pages 367–392. Springer Verlag, 1995.

[KNRS07] Khoussainov, B., Nies, A., Rubin S., and Stephan F., Automatic structures: richness and limitations, *Logical Methods of Computer Science* 3(2), 18 pp. (electronic), 2007.

[MT99] Marek, V.W., and Truszczyński, M. Stable Models and an Alternative Logic Programming Paradigm. *The Logic Programming Paradigm*, pp. 375–398. Series Artificial Intelligence, Springer-Verlag, 1999.

[Nie99] Niemelä, I. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3,4, 241–273, 1999.

[Rog87] Rogers, H. *Theory of Recursive Functions and Effective Computability*, MIT Press, 1987.

[Sch78] Schaefer, T.J. The Complexity of Satisfiability Problems. *Proceedings of STOC 1978*, pages 216–226, 1978.

[Sch98] Schrijver, A. *Combinatorial Optimization*, Springer-Verlag, 2003.