Non-dichotomies in Constraint Satisfaction Complexity

Manuel Bodirsky[1] and Martin Grohe[2]

[1] École Polytechnique (CNRS), France
[2] Humboldt-Universität zu Berlin, Germany

Abstract. We show that every computational decision problem is polynomial-time equivalent to a constraint satisfaction problem (CSP) with an infinite template. We also construct for every decision problem L an *ω-categorical* template Γ such that L reduces to CSP(Γ) and CSP(Γ) is in coNP^L (i.e., the class coNP with an oracle for L). CSPs with ω-categorical templates are of special interest, because the universal-algebraic approach can be applied to study their computational complexity.
Furthermore, we prove that there are ω-categorical templates with coNP-complete CSPs and ω-categorical templates with coNP-intermediate CSPs, i.e., problems in coNP that are neither coNP-complete nor in P (unless P=coNP). To construct the coNP-intermediate CSP with ω-categorical template we modify the proof of Ladner's theorem. A similar modification allows us to also prove a non-dichotomy result for a class of left-hand side restricted CSPs, which was left open in [10]. We finally show that if the so-called *local-global conjecture* for *infinite constraint languages* (over a finite domain) is false, then there is no dichotomy for the constraint satisfaction problem for infinite constraint languages.

1 Introduction

Let Γ be a relational structure over a finite signature τ, also called *template* or *constraint language* in the following. The *constraint satisfaction problem (CSP)* of Γ, denoted by CSP(Γ), is the computational problem to decide whether there exists a homomorphism from a given finite τ-structure to Γ. For finite relational structures Γ it has been conjectured that the computational complexity of CSP(Γ) exhibits a *dichotomy*: it is in P or NP-complete [6, 9].

Ladner's theorem [14] states that there are computational problems in NP that are neither in P nor NP-complete (unless P=NP); we also say that these problems are *NP-intermediate*. However, the problems constructed by Ladner are highly artificial, and the lack of natural computational problems that could be NP-intermediate is one of the phenomena in complexity theory that is not well understood. Even for well-studied problems in NP that are neither known to be in P nor known to be NP-hard, such as the Graph Isomorphism problem, there is usually no strong evidence that they are not in P. From the computational complexity perspective, it would therefore be very interesting if there were NP-intermediate CSPs with finite domains (since CSPs for finite templates are, arguably, relatively natural computational problems).

The CSP dichotomy conjecture is wide open. Bulatov, Krokhin, and Jeavons [6] give a sufficient condition for a finite domain CSP to be NP-hard, and they conjecture that all other problems are in P. This conjecture is based on the so-called *universal-algebraic approach* to the CSP, and the mentioned condition is formulated in terms of an algebra that can be associated to the template Γ. Most work in the area goes into finding larger and larger tractable classes of CSPs [3, 4, 13]. But not all researchers in the area believe in the conjecture. On the negative side, Feder and Vardi [9] introduced several heavily restricted subclasses of NP –for example monadic strict NP (MSNP)– that do *not* have a dichotomy. In fact, every problem in NP has a polynomial-time equivalent problem in MSNP.

In this paper we are mainly interested in infinite templates Γ. A particularly interesting class of infinite templates is the class of ω-*categorical*[3] structures, because ω-categorical templates Γ generalize finite templates in such a way that the above mentioned universal-algebraic approach still applies. For example, it holds that the complexity of the CSP(Γ) is fully captured by the polymorphism clone of Γ (even by the pseudo-variety generated by the algebra of the polymorphism clone [2]).

Results

We show that for every computational decision problem L there exists a polynomial-time equivalent constraint satisfaction problem with an infinite template Γ. This improves previous complexity results about infinite domain constraint satisfaction problems obtained by Bauslaugh [1] and Schwandtner [17]. We also construct an ω-categorical template Γ such that L reduces to CSP(Γ) and CSP(Γ) is in coNP^L (i.e., the class coNP with an oracle for L).

Furthermore, we prove that there are ω-categorical templates with coNP-complete CSPs and ω-categorical templates with *coNP-intermediate* CSPs, i.e., problems in coNP that are neither coNP-complete nor in P (unless P=coNP). To show this we use templates that are *countable homogeneous directed graphs*. These graphs are ω-categorical, and even though there are uncountably many non-isomorphic countable homogeneous directed graphs, they are model-theoretically well-understood [8]. In our construction we apply a modification of Ladner's proof technique. It remains open whether there are ω-categorical structures Γ such that CSP(Γ) is NP-intermediate.

In another line of research, the complexity of the constraint satisfaction problem has been studied for restricted classes of input structures. Let τ be a finite relational signature, and let $\mathcal{C}$ and $\mathcal{D}$ be two classes of finite τ-structures. Then CSP($\mathcal{C}, \mathcal{D}$) is the computational problem where the input consists of *two* structures $C \in \mathcal{C}$ and $D \in \mathcal{D}$, and the question is whether there exists a homomorphism from C to D. It has been shown that when $\mathcal{D}_0$ is the set of all finite τ-structures, then for any recursively enumerable class of finite τ-structures $\mathcal{C}$ the problem CSP($\mathcal{C}, \mathcal{D}_0$) is tractable if and only if $\mathcal{C}$ has bounded tree-width modulo homomorphic equivalence [10]. Note that this is not a dichotomy theorem.

With a similar modification of Ladner's proof as in our previous result, we can show that there is an efficiently decidable class $\mathcal{C}$ of finite (undirected) graphs such that CSP($\mathcal{C}, \mathcal{D}_0$) is NP-intermediate (here, D_0 is the class of all finite undirected graphs); this was left open in [10]. In particular, this shows that there is no dichotomy for problems of the form CSP($\mathcal{C}, \mathcal{D}$). The same result was independently obtained in a recent paper by Chen, Thurley, and Weyer [7].

[3] A structure is ω-categorical if it has for all k only a finite number of *k-types* (in the model-theoretic sense [12]).

Finally, we show a connection between the dichotomy conjecture for *infinite constraint languages over a finite domain* (implied by Conjecture 4.12 of [5]) and the so-called *local-global conjecture*. If Γ is a relational structure with an infinite signature and a finite domain, then $\mathrm{CSP}(\Gamma)$ is called *locally tractable* if $\mathrm{CSP}(\Gamma')$ is tractable for all reducts Γ' of Γ with a finite signature. Note that for the notion of local tractability we can fix an arbitrary representation of the relations in the input instances. $\mathrm{CSP}(\Gamma)$ is called *globally tractable* if there is a polynomial-time algorithm that solves $\mathrm{CSP}(\Gamma)$ where the relations in the input instances are represented by fully specifying the relation $R \subset D(\Gamma)^k$. Obviously, global tractability implies local tractability. The *local-global conjecture* says that $\mathrm{CSP}(\Gamma)$ is locally tractable if and only if it is globally tractable.

If the local-global conjecture is true, then the dichotomy for finite constraint languages implies the dichotomy for infinite constraint languages. We show that in a certain sense the converse is true as well: if the global tractability conjecture is false, then there is no dichotomy for infinite constraint languages. In other words, if the dichotomy conjecture for finite constraint languages is true, then local and global tractability are equivalent.

Preliminaries

A *relational signature* τ is a set of *relation symbols* R_i, each of which has an associated finite *arity* k_i. The signature will be finite unless stated otherwise. A *relational structure* Γ over the signature τ (also called τ-*structure*) consists of a set D_Γ (the *domain*) together with a relation $R \subseteq D_\Gamma^{k_i}$ for each relation symbol of arity k_i from τ. If the reference to the relational structure is clear from the context, we use for simplicity the same symbol for a relation symbol and the corresponding relation. If necessary, we write R^Γ for the relation R belonging to the structure Γ. We call the elements of D_Γ the *vertices* of Γ.

Let Γ and Γ' be τ-structures. A *homomorphism* from Γ to Γ' is a function f from D_Γ to $D_{\Gamma'}$ such that for each n-ary relation symbol R in τ and each n-tuple $(a_1, \ldots, a_n)$, if $(a_1, \ldots, a_n) \in R^\Gamma$, then $(f(a_1), \ldots, f(a_n)) \in R^{\Gamma'}$. In this case we say that the map f *preserves* the relation R. Injective homomorphisms that also preserve the complement of each relation are called *embeddings*. Surjective embeddings are called isomorphisms, and isomorphisms between Γ and Γ are called *automorphisms*.

The *order* of a structure Γ, denoted by $|\Gamma|$, is the cardinality of D_Γ. An *induced substructure* of a structure Γ is a structure Γ' with $D_{\Gamma'} \subseteq D_\Gamma$ and $R^{\Gamma'} = R^\Gamma \cap D_{\Gamma'}^n$ for each n-ary $R \in \tau$. The *union* of two τ-structures Γ, Γ' is the structure $\Gamma \cup \Gamma'$ with domain $D_\Gamma \cup D_{\Gamma'}$ and relations $R^{\Gamma \cup \Gamma'} = R^\Gamma \cup R^{\Gamma'}$ for all $R \in \tau$. The intersection $\Gamma \cap \Gamma'$ is defined similarly. A *disjoint union* of Γ and Γ' is the union of isomorphic copies of Γ and Γ' with disjoint domains. As disjoint unions are unique up to isomorphism, we usually speak of *the* disjoint union of Γ and Γ'. A structure is called *connected* if it is not the disjoint union of two nonempty structures. For a mapping f defined on the domain of a τ-structure Γ, we let $f(\Gamma)$ be the τ-structure with domain $f(D_\Gamma)$ and relations $R^{f(\Gamma)} = \{(f(a_1), \ldots, f(a_n)) \mid (a_1, \ldots, a_n) \in R^\Gamma\}$ for every n-ary $R \in \tau$.

Classes of structures are always assumed to be closed under isomorphism.

2 Templates of All Complexities

In this section we show that for every computational decision problem there exists a polynomial-time equivalent constraint satisfaction problem with an infinite template Γ.

Previously, it was known that for every recursive funtion f there exists an infinite structure Γ such that CSP(Γ) is decidable, but has time complexity at least f (a result due to Bauslaugh [1]). Recently, Schwandtner gave upper and lower bounds in the exponential time hierarchy for some infinite domain CSPs [17]; but these bounds leave an exponential gap.

Theorem 1. *Let L be a language over a finite alphabet Σ. Then there is an infinite relational structure Γ such that L is polynomial-time Turing equivalent to CSP(Γ).*

Proof (Idea). We construct an infinite relational structure Γ over the signature τ that contains pairwise distinct unary relation symbols P_a for all elements of $a \in \Sigma$; moreover, τ contains a binary relation symbol N and unary relation symbols S and T.

For each word $w \in \Sigma^*$ let W be the following τ-structure with vertices $1, \dots, |w|$. The relation N^W is $\{(i, i+1) \mid 1 \leq i < |w|\}$. The unary relation symbol P_a^W holds on $j \in \{1, \dots, |w|\}$ iff the jth symbol in w is a. Finally, $S^W = \{1\}$ and $T^W = \{|w|\}$. Let $\mathcal{A}$ be $\{W \mid w \in L\}$.

Let $\mathcal{X}$ be the set of τ-structures with domain $\{1, \dots, n\}$, for some n, where the symbol N is interpreted by the relation $\{(i, i+1) \mid 1 \leq i < n\}$, each vertex i is contained in at most one relation P_a, for $a \in \Sigma$, and at least one of the following conditions is satisfied:

- S holds for none of the elements $1, \dots, n$.
- T holds for none of the elements $1, \dots, n$.
- There is an element from $1, \dots, n$ such that for all $a \in \Sigma$ the relation P_a does not hold.

The structure Γ is the infinite disjoint union over all structures in $\mathcal{A} \cup \mathcal{X}$. The proof that Γ has the required properties has been omitted due to space restrictions. □

3 ω-categorical Templates of Various Complexities

A relational structure is called *ω-categorical* if for all $k \geq 1$ there are at most finitely many inequivalent first-order formulas with k free variables. This definition of ω-categoricity is by the theorem of Ryll-Nardzewski equivalent to the standard definition of ω-categoricity [12]. With the definition given here it is easy to see that the structure Γ constructed in Section 2 is *not* ω-categorical, since there are infinitely many inequivalent first-order formulas with two free variables x and y that express that x and y are at distance k with respect to the relation N.

In this section we study the computational complexity of CSP(Γ) if Γ is an ω-categorical structure, and prove the following.

Theorem 2. *For every language L over a finite signature Σ there exists an ω-*categorical *structure Γ such that L reduces to CSP(Γ) and CSP(Γ) is in coNP^L.*

For constructing ω-categorical templates, we need a few preliminaries from model theory. Let B_1, B_2 be τ-structures such that $A = B_1 \cap B_2$ is an induced substructure of both B_1 and B_2. Then we call $B_1 \cup B_2$ the *free amalgam* of B_1, B_2 over A. More generally, a τ-structure C is an *amalgam of B_1 and B_2 over A* if for $i = 1, 2$ there are embeddings f_i of B_i to C such that $f_1(a) = f_2(a)$ for all $a \in D_A$. Recall that classes of structures are always assumed to be closed under isomorphism. A class $\mathcal{A}$ of τ-structures has the *amalgamation property* if for all $A, B_1, B_2 \in \mathcal{A}$ with $A = B_1 \cap B_2$ there is a $C \in \mathcal{A}$ that is an amalgam of B_1 and B_2 over A. A class $\mathcal{K}$ of finite τ-structures that has the amalgamation property and is closed under taking induced substructures is called an *amalgamation class.*

The following basic result is known as Fraïssé's Theorem in model theory (see Theorem 6.1.2 in [12]):

Fact 3. *Let $\mathcal{K}$ be an amalgamation class. Then there is an ω-categorical τ-structure Γ such that $\mathcal{K}$ is the class of finite induced substructures of Γ.*

The structure Γ, which is unique up to isomorphism, is called the Fraïssé limit *of $\mathcal{K}$.*

A few remarks are necessary to relate our version of the theorem to the more general version stated, for example, in [12]: Firstly, the amalgamation property is usually defined in a slightly more complicated way where the structure A is not necessarily an induced substructure of the B_i, but embedded into B_i. As we assume classes of structures to be closed under isomorphism, this makes no difference. Secondly, the class $\mathcal{A}$ in Fraïssé's Theorem usually has to have another property known as the *joint embedding property.* However, for relational structures the joint embedding property is subsumed by the amalgmation property. And thirdly, in general the Fraïssé limit Γ is *homogeneous*, but not necessarily ω-categorical. But for finite relational vocabularies, homogeneity implies ω-categoricity.

Let us now turn to the proof of Theorem 2. We encode words over the alphabet Σ by structures similarly, but not exactly as in the proof of Theorem 1. Let τ be a signature that contains the binary relation symbols $N, \neq$, the unary relation symbols S, T, and a unary relation symbol P_a for each $a \in \Sigma$. With each word $w = a_1 \dots a_n \in \Sigma^*$ we associate the τ-structure W with universe $\{0, 1, \dots, n+1\}$, $N^W = \{(i, i+1) \mid 0 \leq i < n+1\}$, $\neq^W = \{(i,j) \mid 0 \leq i, j \leq n+1, i \neq j\}$, $S^W = \{0\}$, $T^W = \{n+1\}$, and $P_a = \{i \mid 1 \leq i \leq n, a_i = a\}$.

For every τ-structure A, we define an undirected graph $G_N(A)$ to be the graph with vertex set D_A and an edge between $a, b \in D_A$ if and only if $a \neq b$ and $(a,b) \in N^A$ or $(b,a) \in N^A$. We say that A is *connected by N* if the graph $G_N(A)$ is connected.

Let $\mathcal{C}$ be the class of all τ-structures isomorphic to a structure W for some $w \in \Sigma^* \setminus L$. Let $\mathcal{K}$ be the class of all τ-structures A with the following properties.

(1) The binary relation $\neq^A$ is *anti-reflexive*, i.e., it does not contain pairs of the form (x,x).
(2) The relations S^A, T^A, and P_a for $a \in \Sigma$ are pairwise disjoint.
(3) N^A is anti-reflexive and anti-symmetric. Furthermore, if $\neq^A$ has cardinality $|A|(|A|-1)$, then the graph $G_N(A)$ is acyclic.
(4) A does not contain a structure from $\mathcal{C}$ as an induced substructure.

Lemma 1. *The class $\mathcal{K}$ is an amalgamation class.*

Proof. It is straightforward to verify that $\mathcal{K}$ is closed under isomorphisms and induced substructures. To show that $\mathcal{K}$ has the amalgamation property, let $B_1, B_2 \in \mathcal{K}$ such that $A = B_1 \cap B_2$ is an induced substructure of both B_1 and B_2. We claim that the free amalgam $C = B_1 \cup B_2$ is contained in $\mathcal{K}$.

We have to prove that C has properties (1)–(4) from page 5. This is obvious for properties (1) and (2), because both B_1 and B_2 have the property and their intersection

A is an induced substructure of both structures. To see that C has property (3), note that either $C = B_1$ or $C = B_2$, and C inherits the property from the respective structure, or $D_C \setminus D_{B_1} \neq \emptyset$ and $D_C \setminus D_{B_2} \neq \emptyset$, and $\neq^C = \neq^{B_1} \cup \neq^{B_2}$ has cardinality less than $|C|(|C|-1)$. It remains to prove that C has property (4). Suppose for contradiction that C has an induced substructure $W \in \mathcal{C}$. As $\neq^W$ connects all pairs of distinct vertices of W, the structure W must be an induced substructure of B_1 or of B_2, which is a contradiction.

Let Γ be the Fraïssé-limit of $\mathcal{K}$.

Lemma 2. *A finite structure A has a homomorphism to Γ if and only if A has properties (1)–(4).*

Proof (of Theorem 2). There is the following reduction from L to CSP(Γ). Given a word $w \in \Sigma^*$, let W be the corresponding τ-structure. If $w \notin L$, then W is in $\mathcal{C}$, and hence W does not satisfy 4. Lemma 2 then implies that there is no homomorphism from W to Γ. If $w \in L$, then $W \notin \mathcal{C}$. The structure W has no induced substructure from $\mathcal{C}$, because every proper induced substructure of W either does not have an element in S, or does not have an element in T, or is not connected by N. Therefore, W homomorphically maps to Γ. This shows that the function that returns for a given word w its word-structure W is a polynomial-time many-one reduction from L to CSP(Γ).

We finally show that CSP(Γ) can be decided by a universal-nondeterministic polynomial-time algorithm with an oracle for L, by the following algorithm.

```
Input: A

If A does not have properties (1)–(3) then reject.
For all induced substructures $W$ of $A$:
    // (can be implemented non-deterministically)
    If $W$ is the $\tau$-structure of a word $w \in \Sigma^*$ then
        If $w \notin L$ then reject.
Accept.
```

Lemma 2 shows that A homomorphically maps to Γ if and only if A satisfies (1)–(4), and this is what the algorithm tests. The algorithm can be implemented on a non-deterministic polynomial-time Turing machine such that there exists a run on input A where the algorithm rejects if and only if A homomorphically maps to Γ.

4 coNP-intermediate ω-categorical Templates

In this section we construct an ω-categorical directed graph Γ such that CSP(Γ) is in coNP, but neither coNP-complete nor in P (unless coNP=P). As in the previous section, the infinite structures studied here are all defined as Fraïssé limits. All structures in this section will be directed graphs.

Henson [11] used Fraïssé limits to construct 2^ω many ω-categorical directed graphs. If $\mathcal{N}$ is a class of τ-structures, *Forb*($\mathcal{N}$) denotes the class of all finite τ-structures A such that no structure from $\mathcal{N}$ embeds into A. A *tournament* is a directed graph G (without self-loops) such that for all pairs x, y of distinct vertices exactly one of the pairs (x, y), (y, x) is an arc in G. Note that for all classes $\mathcal{N}$ of finite tournaments, *Forb*($\mathcal{N}$) is an amalgamation class, because if G_1 and G_2 are directed graphs in $\mathit{Forb}(\mathcal{N})$ such that $H = G_1 \cap G_2$ is an induced substructure of both G_1 and G_2, then the free amalgam $G_1 \cup G_2$ is also in $\mathit{Forb}(\mathcal{N})$. We write $\Gamma_\mathcal{N}$ for the Fraïssé-limit of $\mathit{Forb}(\mathcal{N})$. Observe that for finite $\mathcal{N}$ the problem CSP($\Gamma_\mathcal{N}$) can be solved in deterministic polynomial time, because for a given instance S of this problem an algorithm simply has to check whether there is a homomorphism from one of the structures in $\mathcal{N}$ to S, which is the case if and only if there is a homomorphism from S to $\Gamma_\mathcal{N}$.

Henson in his proof specified an infinite set $\mathcal{T}$ of tournaments $T_1, T_2, \ldots$ with the property that T_i does not embed into T_j if $i \neq j$. Note that this implies that for two distinct subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ of $\mathcal{T}$ the two sets $\mathit{Forb}(\mathcal{T}_1)$ and $\mathit{Forb}(\mathcal{T}_2)$ are distinct as well. Since there are 2^ω many subsets of the infinite set $\mathcal{T}$, there are also that many distinct ω-categorical directed graphs. The tournament T_n in Henson's set $\mathcal{T}$ has vertices $0, \ldots, n+1$, and the following edges:

- (i, j) for $j = i + 1$ and $0 \leq i \leq n$;
- $(0, n + 1)$;
- (j, i) for $j > i + 1$ and $(i, j) \neq (0, n + 1)$.

Proposition 1. *The problem CSP($\Gamma_\mathcal{T}$) is coNP-complete.*

Proof. The problem is contained in coNP, because we can efficiently test whether a sequence $v_1, \ldots, v_k$ of distinct vertices of a given directed graph G induces T_k in G, i.e., whether (v_i, v_j) is an arc in G if and only if (i, j) is an arc in T_k, for all $i, j \in \{1, \ldots, k\}$. If for all such sequences of vertices this test is negative, we can be sure that G is from $\mathit{Forb}(\mathcal{T})$, and hence homomorphically maps to $\Gamma_\mathcal{T}$. Otherwise, G embeds a structure from $\mathcal{T}$, and hence does not homomorphically map to $\Gamma_\mathcal{T}$.

The proof of coNP-hardness goes by reduction from the complement of the NP-complete 3-SAT problem, and is inspired by a classical reduction from 3-SAT to Clique. For a given 3-SAT instance, we create an instance G of CSP($\Gamma_\mathcal{T}$) as follows: If

$$\{x_0^1, x_0^2, x_0^3\}, \ldots, \{x_{k+1}^1, x_{k+1}^2, x_{k+1}^3\}$$

are the clauses of the 3-SAT formula (we assume without loss of generality that the 3-SAT instance has at least three clauses), then the vertex set of G is $\{(0,1), (0,2), (0,3), \ldots, (k+1,1), (k+1,2), (k+1,3)\}$, and the arc set of G consists of all pairs $((i,j),(p,q))$ of vertices such that $x_i^j \neq \neg x_p^q$ and such that (i, p) is an arc in T_k.

We claim that a 3-SAT instance is unsatisfiable if and only if the created instance G homomorphically maps to $\Gamma_\mathcal{T}$. The 3-SAT instance is satisfiable iff there is a mapping from the variables to true and false such that in each clause at least one literal, say $x_0^{j_0}, \ldots, x_{k+1}^{j_{k+1}}$, is true. This is the case if and only if the vertices $(0, j_1), \ldots, (k+1, j_{k+1})$ induce T_k in G, i.e., $((i, j_i), (p, j_p))$ is an edge if and only if (i, p) is an edge in T_k. This is the case if and only if T_k embeds into G, and if and only if G does not homomorphically map to $\Gamma_\mathcal{T}$. □

We now modify the proof of Ladner's Theorem given in [16] (which is basically Ladner's original proof) to create a subset $\mathcal{T}_0$ of $\mathcal{T}$ such that CSP($\Gamma_{\mathcal{T}_0}$) is in coNP, but neither in P nor coNP-complete (unless coNP=P). One of the ideas in Ladner's proof is to *'blow holes into SAT'*, such that the resulting problem is too sparse to be NP-complete and to dense to be in P. Our modification is that we do not blow holes into a computational problem itself, but that we *'blow holes into the obstruction set $\mathcal{T}$ of CSP($\Gamma_{\mathcal{T}}$)'*.

In the following, we fix one of the standard encodings of graphs as strings over the alphabet $\{0,1\}$. Let $M_1, M_2, \ldots$ be an enumeration of all polynomial-time bounded Turing machines, and let $R_1, R_2, \ldots$ be an enumeration of all polynomial time bounded reductions. We assume that these enumerations are effective; it is well-known that such enumerations exist.

The definition of $\mathcal{T}_0$ uses a Turing machine F that computes a function $f : \mathbb{N} \to \mathbb{N}$, which is defined below. The set $\mathcal{T}_0$ is then defined as follows.

$$\mathcal{T}_0 = \{T_n \mid f(n) \text{ is even }\}$$

The input number n is given to the machine F in unary representation. The computation of F proceeds in two phases. In the first phase, F simulates itself[4] on input 1, then on input 2, 3, and so on, until the number of computation steps of F in this phase exceeds n (we can always maintain a counter during the simulation to recognize when to stop). Let k be the value $f(i)$ for the last input i for which the simulation was completely performed by F.

In the second phase, the machine stops if phase two takes more than n computation steps, and F returns k. We distinguish whether k is even or odd. If k is even, all directed graphs G on $s = 1, 2, 3, \ldots$ vertices are enumerated. For each directed graph G in the enumeration the machine F simulates $M_{k/2}$ on the encoding of G. Moreover, F computes whether G homomorphically maps to $\Gamma_{\mathcal{T}_0}$. This is the case if for all structures $T_l \in \mathcal{T}$ that embed into G the value of $f(l)$ is even. So F tests for $l = 1, 2, \ldots, s$ whether T_l embeds to G (F uses any straightforward exponential time algorithm for this purpose), and if it does, simulates itself on input l to find out whether $f(l)$ is even. If

(1) $M_{k/2}$ rejects and G homomorphically maps to $\Gamma_{\mathcal{T}_0}$, or
(2) $M_{k/2}$ accepts and G does not homomorphically map to $\Gamma_{\mathcal{T}_0}$,

then F returns $k+1$ (and $f(n) = k+1$).

The other case of the second phase is that k is odd. Again F enumerates all directed graphs G on $s = 1, 2, 3, \ldots$ vertices, and simulates the computation of $R_{\lfloor k/2 \rfloor}$ on the encoding of G. Then F computes whether the output of $R_{\lfloor k/2 \rfloor}$ encodes a directed graph G' that homomorphically maps to $\Gamma_{\mathcal{T}_0}$. The graph G' homomorphically maps to $\Gamma_{\mathcal{T}_0}$ iff for all tournaments T_l that embed into G' the value $f(l)$ is even. Whether T_l embeds into G' is tested with a straightforward exponential-time algorithm. To test whether $f(l)$ is even, F simulates itself on input l. Finally, F tests with a straightforward exponential-time algorithm whether G homomorphically maps to $\Gamma_{\mathcal{T}}$. If

(3) G homomorphically maps to $\Gamma_{\mathcal{T}}$ and G' does not homomorphically map to $\Gamma_{\mathcal{T}_0}$, or
(4) G does not homomorphically map to $\Gamma_{\mathcal{T}}$ and G' homomorphically maps to $\Gamma_{\mathcal{T}_0}$,

then F returns $k+1$.

Lemma 3. *The function f is a non-decreasing function, that is, for all n we have $f(n) \leq f(n+1)$.*

[4] Note that by the fixpoint theorem of recursion theory we can assume that F has access to its own description.

Lemma 4. *For all n_0 there is $n > n_0$ such that $f(n) > f(n_0)$ (unless $coNP \neq P$).*

Proof. Assume for contradiction that there exists an n_0 such that $f(n)$ equals a constant k_0 for all $n \geq n_0$. Then there also exists an n_1 such that for all $n \geq n_1$ the value of k computed by the first phase of F on input n is k_0.

If k_0 is even, then on all inputs $n \geq n_1$ the second phase of F simulates $M_{k_0/2}$ on encodings of an enumeration of graphs. Since the output of F must be k_0, for all graphs neither (1) nor (2) can apply. Since this holds for all $n \geq n_1$, the polynomial-time bounded machine $M_{k_0/2}$ correctly decides CSP($\Gamma_{\mathcal{T}_0}$), and hence CSP($\Gamma_{\mathcal{T}_0}$) is in P. But then there is the following polynomial-time algorithm that solves CSP($\Gamma_{\mathcal{T}}$), a contradiction to coNP-completeness of CSP($\Gamma_{\mathcal{T}}$) (Proposition 1) and our assumption that coNP $\neq$ P.

```
Input: A directed graph $G$.

Test whether $G$ homomorphically maps to $\Gamma_{\mathcal{T}_0}$.
If yes, accept.
If no, test whether one of the finitely many graphs in $\mathcal{T} \setminus \mathcal{T}_0$ embeds into $G$.
Accept if none of them embeds into $G$.
Reject otherwise.
```

If k_0 is odd, then on all inputs $n \geq n_1$ the second phase of F does not find a graph G for which (3) or (4) applies, because the output of F must be k_0. Hence, $R_{\lfloor k_0/2 \rfloor}$ is a polynomial-time reduction from CSP($\Gamma_{\mathcal{T}}$) to CSP($\Gamma_{\mathcal{T}_0}$), and by Proposition 1 the problem CSP($\Gamma_{\mathcal{T}_0}$) is coNP-hard. But note that because $f(n)$ equals the odd number k_0 for all but finitely many n, the set $\mathcal{T}_0$ is finite. Therefore, CSP($\Gamma_{\mathcal{T}_0}$) can be solved in polynomial time, contradicting our assumption that coNP $\neq$ P. □

Theorem 4. *CSP($\Gamma_{\mathcal{T}_0}$) is in coNP, but neither in P nor coNP-complete (unless coNP=P).*

Proof. It is easy to see that CSP($\Gamma_{\mathcal{T}_0}$) is in coNP. On input G the algorithm non-deterministically chooses a sequence of l vertices, and checks in polynomial time whether this sequence induces a copy of T_l. If yes, the algorithm computes $f(l)$, which can be done in linear time by executing F on the unary representation of l. If $f(l)$ is even, the algorithm accepts. Recall that G does not homomorphically map to $\Gamma_{\mathcal{T}_0}$ iff a tournament $T_l \in \mathcal{T}_0$ embeds into G, which is the case iff there is an accepting computation path for the above non-deterministic algorithm.

Suppose that CSP($\Gamma_{\mathcal{T}_0}$) is in P. Then for some i the machine M_i decides CSP($\Gamma_{\mathcal{T}_0}$). By Lemma 3 and Lemma 4 there exists an n_0 such that $f(n_0) = 2i$. Then there must also be an $n_1 > n_2$ such that the value k computed during the first phase of F on input n_1 equals $2i$. Since M_i correctly decides CSP($\Gamma_{\mathcal{T}_0}$), the machine F returns $2i$ on input n_1. By Lemma 3, the machine F also returns $2i$ for all inputs from n_1 to n_2, and by induction it follows that it F returns $2i$ for *all* inputs larger than $n \geq n_0$, in contradiction to Lemma 4.

Finally, suppose that CSP($\Gamma_{\mathcal{T}_0}$) is coNP-complete. Then for some i the machine R_i is a valid reduction from $CSP(\Gamma_{\mathcal{T}})$ to CSP($\Gamma_{\mathcal{T}_0}$). Again, by Lemma 3 and Lemma 4 there exists an n_1 such that the value k computed during the first phase of F on input n_1 equals $2i$. Since the reduction R_i is correct, the machine F returns $2i$ on input n_1, and in fact returns $2i$ on all inputs greater than n_1. This contradicts Lemma 4. □

5 Left-hand Side Restrictions

Let $\mathcal{S}$ be a class of finite τ-structures. Then CSP($\mathcal{S}$, _) is the computational problem to decide whether for a given pair (A, B) of finite τ-structures with $A \in \mathcal{S}$ there is a homomorphism from A to B.

As an example, let τ be the signature that consists of a single binary relation. In this case, τ-structures can be considered as directed graphs (potentially with loops). If $\mathcal{C}$ is the set of all complete graphs (without loops!), then CSP($\mathcal{C}$, _) is essentially a formulation of the Clique problem.

The following question was left open in [10]: *Are there classes of structures $\mathcal{S}$ such that CSP($\mathcal{S}$, _) is in NP, but neither in P nor NP-complete?*

We answer this question positively, and construct such a class $\mathcal{S}$, which can even be decided in polynomial time. Again we use a modification of Ladner's theorem. The modification is similar to the modification presented in Section 4. This time, we *'blow holes into the possible clique sizes for the clique problem'* and obtain a class $\mathcal{C}_0 \subseteq \mathcal{C}$ such that CSP($\mathcal{C}_0$, _) is in NP $\setminus$ P and not NP-complete (unless P=NP).

The idea is to define $\mathcal{C}_0$ in such a way that the $\mathcal{C} \setminus \mathcal{C}_0$ becomes finite when CSP($\mathcal{C}_0$, _) is in P; hence, CSP($\mathcal{C}_0$, _) is polynomial-time equivalent to the Clique problem, a contradiction unless P=NP. Moreover, the construction of $\mathcal{C}_0$ is such that $\mathcal{C}_0$ is finite if CSP($\mathcal{C}_0$, _) is NP-hard. But for finite $\mathcal{C}_0$, the problem CSP($\mathcal{C}_0$, _) is in P, again contradicting the assumption that P $\neq$ NP. We also take extra care to make $\mathcal{C}_0$ polynomial-time decidable.

Theorem 5. *CSP($\mathcal{C}_0$, _) is in NP, but neither in P nor NP-complete (unless P=NP). Moreover, the set $\mathcal{C}_0$ can be decided in deterministic polynomial time.*

In our proof we do not use any specific properties of the class $\mathcal{C}$ and the clique problem, but in fact we can construct classes $\mathcal{S}_0 \subseteq \mathcal{S}$ with NP-intermediate CSP($\mathcal{S}_0$, _) for any class $\mathcal{S}$ where CSP($\mathcal{S}$, _) is NP-complete.

6 The Local-Global Conjecture

The complexity of the constraint satisfaction problem has also been studied for templates Γ with an *infinite signature*. Several well-known computational problems can be modeled as CSPs only if we allow (countably) infinite constraint languages: examples are boolean Horn-satisfiability, Ord-Horn constraints in temporal reasoning [15], or solving linear equation systems over a finite field.

If the local-global conjecture as stated in Section 1 is true, then the dichotomy for finite constraint languages implies the dichotomy for infinite constraint languages: if an infinite constraint language Γ has a finite reduct Γ' such that CSP(Γ') is NP-hard, then CSP(Γ) is clearly NP-hard as well. On the other hand, if CSP(Γ) is locally tractable, then the conjecture implies that CSP(Γ) is globally tractable.

We show that if the local-global conjecture is false, then there is no dichotomy for infinite constraint languages.

Theorem 6. *If the local-global conjecture is false, then there exists a template Γ_0 with finite domain D and infinite signature such that CSP(Γ_0) is neither globally tractable nor NP-complete (unless P=NP); moreover, the meta-problem for Γ_0 is efficiently decidable, i.e., given a relation R over D, we can decide in polynomial-time whether R is in Γ_0.*

The proof of Theorem 6 is essentially again a modification of Ladner's theorem, but we have to overcome a complication: for a straightforward application of Ladner's theorem, we need that if $\mathit{\Gamma}$ is an expansion of $\mathit{\Gamma}'$ by finitely many relation symbols, and CSP($\mathit{\Gamma}$) is NP-hard, then CSP($\mathit{\Gamma}'$) is NP-hard as well: but this is not true in general. We only sketch the basic setting of Ladner's construction, which we have already seen twice in this paper, and focus on the complication.

Proof (Sketch). Assume that there is an infinite constraint language $\mathit{\Gamma}$ with a finite domain such that CSP($\mathit{\Gamma}$) is locally tractable, but not globally tractable. If $\mathit{\Gamma}$ is not NP-complete, we are already done with $\mathit{\Gamma}_0 = \mathit{\Gamma}$, so assume that $\mathit{\Gamma}$ is NP-complete. We claim that we can assume without loss of generality that $\mathit{\Gamma}$ contains all primitive positive definable relations (for an introduction to this basic concept in model theory and its applications in constraint satisfaction theory, see e.g. [6]). To see this, first observe that the expansion $\mathit{\Gamma}'$ of $\mathit{\Gamma}$ by all those relations trivially still has an NP-complete CSP. Moreover, all reducts of $\mathit{\Gamma}'$ with a finite signature have a CSP that is in P, because all relations in this reduct can be defined by finitely many relations from $\mathit{\Gamma}$. So, $\mathit{\Gamma}'$ can be obtained from a finite reduct of $\mathit{\Gamma}$ by expansion with finitely many primitive positive relations, and hence CSP($\mathit{\Gamma}'$) is in P.

We construct a reduct $\mathit{\Gamma}_0$ of $\mathit{\Gamma}$ such that CSP($\mathit{\Gamma}_0$) is neither in P nor NP-complete (unless P=NP). Again, the definition of $\mathit{\Gamma}_0$ is by a Turing machine F that computes a function $f : \mathbb{N} \rightarrow \mathbb{N}$, and the n-th relation of $\mathit{\Gamma}$ (according to some fixed enumeration of the relations of $\mathit{\Gamma}$) is in $\mathit{\Gamma}_0$ if $f(n)$ is even.

As in the proofs before, n is given to the machine F in unary representation, and we can define F in such a way that

– it runs in polynomial time in n, and
– $\mathit{\Gamma}_0$ becomes finite if CSP($\mathit{\Gamma}_0$) is NP-hard, and
– $\mathit{\Gamma}$ is an expansion of $\mathit{\Gamma}_0$ by finitely many relations if CSP($\mathit{\Gamma}_0$) is in P.

If $\mathit{\Gamma}_0$ is finite, then CSP($\mathit{\Gamma}_0$) is tractable, because every reduct of $\mathit{\Gamma}$ with a finite signature is by assumption tractable, and we obtain that P=NP. Now, suppose that we are in the other case, and that $\mathit{\Gamma}$ is an expansion of $\mathit{\Gamma}_0$ by finitely many relations. We want to show that CSP($\mathit{\Gamma}_0$) is NP-hard by reducing CSP($\mathit{\Gamma}$) to CSP($\mathit{\Gamma}_0$).

Let S be an instance of CSP($\mathit{\Gamma}$). Note that S might contain constraints for the relations from $\mathit{\Gamma}$ that are not in $\mathit{\Gamma}_0$, but there is a k such that all those constraints have arity less than k. Because $\mathit{\Gamma}$ contains all primitive positive definable relations, $\mathit{\Gamma}$ contains in particular for every l-ary relation R the k-ary relation $R' := R \times D \times \cdots \times D$. We now replace each constraint in S with a relation R from $\mathit{\Gamma}$ that is not in $\mathit{\Gamma}_0$ by a constraint for R', introducing $k - l$ new dummy variables for the last $k - l$ arguments of R'. Even though the representation of R' is much larger than the representation of R, this can only lead to a linear increase in the size of the instance S, because both D and k are fixed. The resulting structure S' is an instance of CSP($\mathit{\Gamma}_0$), and S' homomorphically maps to $\mathit{\Gamma}_0$ if and only if S homomorphically maps to $\mathit{\Gamma}$. Therefore CSP($\mathit{\Gamma}_0$) is NP-complete, which again implies that P=NP. □

Acknowledgements. We would like to thank Hubie Chen for his helpful comments on an earlier version of this paper.

References

1. B. L. Bauslaugh. The complexity of infinite h-coloring. *J. Comb. Theory, Ser. B*, 61(2):141–154, 1994.
2. M. Bodirsky. Constraint satisfaction problems with infinite templates. *Survey, to appear*, 2007.
3. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of LICS'03*, pages 321–330, 2003.
4. A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings of LICS'04, Turku, Finland*, 2004.
5. A. Bulatov and P. Jeavons. Algebraic structures in combinatorial problems. Technical report MATH-AL-4-2001, Technische Universitat Dresden, submitted to International Journal of Algebra and Computing, 2001.
6. A. Bulatov, A. Krokhin, and P. G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.
7. Y. Chen, M. Thurley, and M. Weyer. Understanding the complexity of induced substructure isomorphisms. In *ICALP'08*, 2008.
8. G. Cherlin. The classification of countable homogeneous directed graphs and countable homogeneous n-tournaments. *AMS Memoir*, 131(621), January 1998.
9. T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.
10. M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.
11. C. W. Henson. Countable homogeneous relational systems and categorical theories. *Journal of Symbolic Logic*, 37:494–500, 1972.
12. W. Hodges. *A shorter model theory.* Cambridge University Press, 1997.
13. P. M. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. In *LICS'07*, pages 213–224, 2007.
14. R. E. Ladner. On the structure of polynomial time reducibility. *JACM*, 22(1):155–171, 1975.
15. B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *JACM*, 42(1):43–66, 1995.
16. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
17. G. Schwandtner. Datalog on infinite structures. Submitted dissertation, Humboldt-Universität zu Berlin, 2008.

A Proofs for Section 2

Proof (of Theorem 1). We show that CSP(Γ) is equivalent to L with respect to polynomial-time Turing reductions. We first describe an algorithm that uses an oracle for L and decides CSP(Γ) in polynomial time. Let A be a finite relational τ-structure. If A has an empty domain, there is a homomorphism from A to Γ, and the algorithm accepts. Otherwise, the algorithm performs a breadth-first search (BFS) on the directed graph (possibly with loops) defined by the relation N in A. The BFS starts with some vertex u_0, and labels u_0 with 1. If in the course of the BFS the algorithm detects an edge (u, v) where u (v, respectively) has already been labeled by i, then the vertex v (u, respectively) gets the label $i + 1$ ($i - 1$, respectively). If v (u, respectively) was already labeled by a distinct label in a previous step of the algorithm, there is clearly no homomorphism from A to Γ, and the algorithm rejects. Moreover, if any vertex gets a label smaller than the label of a vertex $u \in S$, or if any vertex gets a label larger than the label of a vertex $u \in T$, then there is no homomorphism from A to Γ, and the algorithm rejects.

Let B be the τ-structure with vertices $1, \dots, l$, for $l = t - s + 1$ where s is the smallest and t the largest label assigned to vertices of A. The relation N is interpreted by $\{(i, j) \mid j = i + 1, 1 \leq i < l\}$, and a unary relation P holds on a vertex $i - s + 1$ in B if there is a vertex u of A such that P holds on u in A and u is labeled with i. Observe that the substructure A' of A that is induced by all vertices that were labeled by the run of the BFS that started at u_0 homomorphically maps to B: the homomorphism maps vertices u with label i to $i - s + 1$. Also observe that any homomorphism from A' to Γ must map two vertices with the same label to the same vertex in Γ. Hence, the homomorphism described above is unique. Moreover, if there are two vertices u, u' with the same label in A' such that P_a holds for u and P_b holds for u' for $a \neq b$, then there is no homomorphism from A' to Γ. In this case, the algorithm rejects. In the following we therefore assume that for each label there is at most one $a \in \Sigma$ such that P_a holds on vertices with this label.

If there is a label such that for all vertices with this label no predicate P_a for $a \in \Sigma$ holds, or if no vertex in A' satisfies S, or if no vertex in A' satisfies T, then B is in $\mathcal{X}$ and therefore A' homomorphically maps to Γ. In this case, the algorithm removes the vertices of A' from A, and proceeds recursively on the remaining graph. Otherwise, it can be verified that B equals W for some word $w \in \Sigma^*$ (in particular, vertex 1 satisfies S and vertex l satisfies T). The algorithm then uses the oracle to decide whether w is in L. If the answer is yes, there is a homomorphism from A' to the copy of W in Γ, and the algorithm removes the vertices of A' from A and proceeds recursively on the remaining graph. Otherwise, the algorithms rejects.

We claim that if the algorithm rejects there is no homomorphism from A' to Γ. Assume otherwise. Since A' is connected, any homomorphism from A' to Γ must map A' to a copy of a structure in $\mathcal{A}$, or a copy of a structure in $\mathcal{X}$. Since vertex 1 satisfies S and vertex l satisfies T, and since for each label $i \in \{s, \dots, t\}$ there is at least one vertex labeled by i and satisfying P_a for some $a \in \Sigma$, there is no homomorphism from A' to structures in $\mathcal{X}$. So lets assume that there is a homomorphism from A' to a structure W in $\mathcal{A}$. In this case, the structure W and the structure B described above must be isomorphic, and the oracle must have answered yes, a contradiction. It is now obvious from the description of the algorithm that it accepts if and only if there is a homomorphism from A to Γ.

Next, we show that there is a polynomial-time many-one reduction from L to $\mathrm{CSP}(\Gamma)$. For a given word w, the reduction returns the structure W described above. It is clear that W is efficiently computable, and that W homomorphically maps to Γ if $w \in L$. If $w \notin L$, we claim that W does not homomorphically map to Γ. If there was a homomorphism from W to Γ, then by construction of Γ all elements must be mapped to the same copy of a structure W for $w \in L$ or to a structure from $\mathcal{X}$, because W is connected. In fact, it can not be that W homomorphically maps to a structure from $\mathcal{X}$, because then the relation S, the relation T, or one of the relations P_a will be violated. Hence, all elements of W must be mapped to the elements of a copy of $A_{w'}$ for some $w' \in L$. It is easy to see that if there is a homomorphism from W to W', then the word w must be equal to w'. Hence, $w \in L$, a contradiction. □

B Proofs for Section 3

Proof (of Lemma 2). If A has properties (1)–(4), then it is in $\mathcal{K}$ and hence embeds into Γ.

Conversely, suppose that h is a homomorphism from A to Γ. Then $\neq^A$ must be anti-reflexive, because otherwise Γ would not be anti-reflexive, in contradiction to the fact that all structures in $\mathcal{K}$ satisfy (1). By similar reasoning, A also has properties (2) and (3).

To show that A also has property (4), suppose for contradiction that A has an induced substructure W from $\mathcal{C}$. Let W' be the induced substructure of Γ with domain $h(D_W)$. Then $W' \in \mathcal{K}$. We claim that the restriction h' of h to D_W is an isomorphism from W to W'. This implies $W' \in \mathcal{C}$, which is a contradiction. To prove the claim, we first note that h' is one-to-one, because $\neq^W$ holds for all distinct elements in W, and $\neq^\Gamma$ is anti-reflexive. Next, we observe that h' preserves the complement of N: As W is connected by N and h' is a homomorphism, W' is also connected by N. Hence if there were $a, b \in D_W$ such that $(a, b) \notin N^W$ and $(h'(a), h'(b)) \in N^{W'}$, then either $N^{W'}$ would not be antireflexive or $N^{W'}$ would not be antisymmetric or $G_N(W')$ would contain a cycle, which contradicts W' having property (3). Finally, h' also preserves the complement of the relations S, T, and P_a for all $a \in \Sigma$, because each vertex of W is contained in precisely one of these relations, and by (2) each vertex of W' is contained in at most one of the relations.

C Proofs for Section 4

Proof (of Lemma 3). We inductively assume that $f(s-1) \leq f(s)$ for all $s \leq n$, and have to show that $f(n) \leq f(n+1)$. Since F has more time to simulate itself when we run it on $n+1$ instead of n, the value i computed in the first phase of F cannot become smaller. By inductive assumption, $k = f(i)$ cannot become smaller as well. In the second phase, we either return k or $k+1$. Hence, if k becomes larger in the first phase, the output of F cannot become smaller. If k does not become larger, then the only difference between the second phase of F for input $n+1$ compared to input n is that there is more time for the computations. Hence, if the machine F on input n verifies condition (1),(2),(3),(4) for some graph G (and hence returns $k+1$), then F also verifies this condition for G on input $n+1$, and returns $k+1$ as well. Otherwise, $f(n) = k$, and also here $f(n+1) \geq f(n)$ holds. □

D Proofs for Section 5

Let $M_1, M_2, \ldots$ and $R_1, R_2, \ldots$ be effective enumerations of all polynomial-time bounded Turing machines, and all logarithmic-space reductions, respectively. For the rest of the proof, we fix any standard encoding of pairs of graphs by words in $\{0,1\}^*$. Again, the definition of $\mathcal{C}_0$ is by a Turing machine F that computes a function $f : \mathbb{N} \to \mathbb{N}$. Let the clique K_n (i.e., the complete graph without loops) be in $\mathcal{C}_0$ iff $f(n)$ is even. The input number n is given to the machine F in unary representation. The computation of F proceeds in two phases. In the first phase, F simulates itself on input $1, 2, \ldots,$ until the number of steps in this phase exceeds n. Let k be the value $f(i)$ for the last value i for which the simulation was completely performed by F.

In the second phase, we distinguish whether k is even or odd. In both cases, the machine stops if phase two takes more than n computation steps, and in this case F returns k. If k is even, F enumerates all pairs of graphs (G, H) where G is a clique and H is an arbitrary graph. For each pair the machine F simulates $M_{k/2}$ on the encoding of (G, H). Moreover, F computes whether G homomorphically maps to H. Finally, F simulates itself to find out whether $f(l)$ is even, where l is the number of vertices of G. If

(1) $M_{k/2}$ rejects, G homomorphically maps to H, and $f(l)$ is even, or
(2) $M_{k/2}$ accepts, G does not homomorphically map to H, and $f(l)$ is even,

then F returns $k + 1$ (and $f(n) = k + 1$).

The other case of the second phase is that k is odd. Again F enumerates all pairs of graphs (G, H), and simulates the computation of $R_{\lfloor n/2 \rfloor}$ on the encoding of (G, H). Then F decides whether the output of $R_{\lfloor n/2 \rfloor}$ encodes a pair (G', H') such that G' is a clique on l vertices, and such that G' homomorphically maps to H; this can be done using any exponential-time algorithm. Finally, F simulates itself on input l to test whether $f(l)$ is even. If

– $f(l)$ is odd, or
– G homomorphically maps to H and
  G' does not homomorphically map to H', or
– G does not homomorphically map to H and
  G' homomorphically maps to H',

then F returns $k + 1$.

As in Section 4, it is straightforward to verify that f is a non-decreasing function, and that f can not be constant for all large n unless P=NP.

Proof (of Theorem 5). The containment in NP follows directly from the containment of the more general problem CSP($\mathcal{C}$,_) in NP. The set $\mathcal{C}_0$ can be decided in deterministic polynomial time, because the function f can be computed in linear time in n.

Suppose that CSP($\mathcal{C}_0$, _) is in P. Then for some i the machine M_i decides CSP($\mathcal{C}_0$, _). By the observations above, there exists an n_0 such that $f(n_0) = 2i$. Then there must also be an $n_1 > n_0$ such that the value k computed during the first phase of F on input n_1 equals $2i$. Since M_i correctly decides CSP($\mathcal{C}_0$, _), the machine F returns $2i$ for all inputs larger than $n \geq n_0$, which contradicts the observation that f cannot be constant for large n.

Finally, suppose that CSP($\mathcal{C}_0$, _) is NP-complete. Then for some i the machine R_i is a valid reduction from CSP($\mathcal{C}$, _) to CSP($\mathcal{C}_0$, _). Again, there exists an n_1 such that the value k computed during the first phase of F on input n_1 equals $2i-1$. Since the reduction R_i is correct, the machine F returns $2i-1$ on input n_1, and in fact returns $2i-1$ on all inputs greater than n_1. Again, this is a contradiction. $\square$