Graph Ramsey Theory and the Polynomial Hierarchy

Marcus Schaefer

School of CTI
DePaul University
243 South Wabash Avenue
Chicago, Illinois 60604, USA
schaefer@cs.depaul.edu

November 27, 2000

Abstract

In the Ramsey theory of graphs $F \to (G, H)$ means that for every way of coloring the edges of F red and blue F will contain either a red G or a blue H. ARROWING, the problem of deciding whether $F \to (G, H)$, lies in $\mathbf{\Pi_2^p} = \mathbf{coNP^{NP}}$ and it was shown to be $\mathbf{coNP}$-hard by Burr [Bur90]. We prove that ARROWING is $\mathbf{\Pi_2^p}$-complete, simultaneously settling a conjecture of Burr and providing a rare natural example of a problem complete for a higher level of the polynomial hierarchy. We also show that STRONG ARROWING, the induced subgraph version of ARROWING, is $\mathbf{\Pi_2^p}$-complete, and that the complexity of not arrowing stars is the same as that of finding a perfect matching.

1 Introduction

Party mathematics is an important tool in the repertoire of the socially gifted mathematician, and one of the all-time favorite stories tells us that at a party of six people there are at least three people who know each other, or three people who do not know each other. When mathematicians were invited to larger parties they began working on the general problem: How many people do you need to have at least k people who know each other, or l people who do not know each other? It is not clear, a priori, that such a number even exists. Frank Ramsey [Ram30, GRS90] showed in 1930 that the number of people needed is finite. Since then Ramsey theory has developed into an active and rich field.

Looking at the party example again, we see that it could also have been phrased thus: every edge-coloring of the complete graph on six vertices K_6 in red and blue contains either a red or a blue triangle. Graphs offer a generalized approach to classical Ramsey theory which over the last twenty years has turned out to be quite fruitful [GRS90]. The basic notion of graph Ramsey theory is arrowing: we say that a graph F *arrows* (G, H) and write $F \to (G, H)$ if for every edge-coloring of F with colors red and blue, a red G or a blue H occurs as a subgraph. We say F *strongly arrows* (G, H) and write $F \rightarrowtail (G, H)$ if for every edge-coloring of F with colors red and blue, a red G or a blue H occurs as an induced subgraph of F.

Take for example $G = H = P_4$, a path on four vertices. One quickly verifies that $K_5 \to (P_4, P_4)$ [CH72]. In the induced case the complete graph will not do, since K_5 only has complete induced subgraphs. However, the graph obtained from the Möbius ladder M_8 (a C_8 in which all pairs of opposite vertices are connected) by removing one of the spokes, strongly arrows P_4 [SS00]. See Figure 1 for a picture of the graph. (Harary, Nešetřil, and Rödl [HNR83] claimed that M_8 itself would do, but this is not the case.)

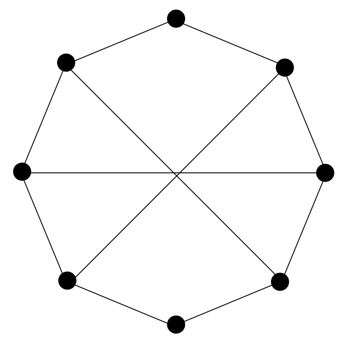

Figure 1: A variant of the Möbius ladder M_8

Given that arrowing is the central notion of graph Ramsey theory, it is natural to ask what its computational complexity is: How hard is it to determine whether $F \to (G, H)$? In the notation of Garey and Johnson we can define two problems:

ARROWING
Instance: (Finite) graphs F, G, and H.
Question: Does $F \to (G, H)$?

STRONG ARROWING
Instance: (Finite) graphs F, G, and H.
Question: Does $F \rightarrowtail (G, H)$?

We will show that both ARROWING and STRONG ARROWING are complete for the second level of the polynomial hierarchy, settling a conjecture of Burr's [Bur87, Bur90], and adding to the rather small collection of problems complete for higher levels of the polynomial hierarchy.

As a consequence of these results several other Ramsey type questions can be reduced (by a polynomial-time algorithm) to a single question $F \to (G, H)$, including Ramsey problems with several colors, or problems of the type $F \to (\mathcal{G}, \mathcal{H})$ (where $\mathcal{G}$ and $\mathcal{H}$ are families of graphs). On the other hand there cannot be a polynomial time algorithm converting a question of the form $F \not\to (G, H)$ to an equivalent question of the form $F' \to (G', H')$ unless the polynomial hierarchy collapses to the second level.

Let us have a look at the history of the ARROWING problem. Fixing the graphs G and H makes $F \to (G, H)$ a **coNP** problem in input F since checking for fixed subgraphs can be done in polynomial time for any coloring. Stefan Burr [Bur90] showed that this problem is complete for **coNP** for any fixed 3-connected graphs G and H (a graph is *3-connected* if it remains connected when any two vertices are removed from it). An earlier version of this result, also by Burr, where $G = H = K_3$ is already mentioned by Garey and Johnson [GJ79, Problem GT6]. Compared to this restricted version, ARROWING immediately appears stronger, since $F \to (P_2, K_n)$ decides whether F has a clique of size n and is therefore **NP**-hard (P_2 is the path on two vertices, that is, a single edge).

The **coNP**-completeness proof of Burr for the restricted version requires the construction of particular graphs called determiners and senders. Although the constructions are effective, they are not feasible, meaning they cannot be performed in polynomial time given G and H as inputs (for the simple reason that the constructed graphs are too large). To overcome this obstacle in the $\mathbf{\Pi_2^p}$-completeness proof we will not use two 3-connected graphs, but a tree and a complete graph.

If we take the ARROWING problem and consider vertex-colorings instead of edge-colorings we get the GENERALIZED GRAPH COLORING problem: does every two-coloring of the vertices of F contain a vertex-monochromatic G? This problem was shown $\mathbf{\Pi_2^p}$-complete by Rutenburg [Rut86], even in the special case that G is restricted to complete graphs (a case that is open for edge-colorings).

We begin the paper with a short survey of completeness results for the second level of the polynomial hierarchy in Section 3. In Section 4 and Section 5 we establish the completeness results for ARROWING and STRONG ARROWING, respectively. In these results the second graph H is always a complete graph. In Section 6 we discuss completeness results for other families of graphs, and in Section 7 we study the complexity of the diagonal case (where $G = H$). In Section 8 we consider arrowing problems which are not $\mathbf{\Pi_2^p}$-complete (unless the hierarchy collapses). Section 9 concludes the paper with a short summary of results, and a discussion of open problems, and other related research.

2 Preliminaries

As general references we suggest Graham, Rothschild, and Spencer on Ramsey theory [GRS90], Diestel on graph theory [Die97], and Garey and Johnson on complexity theory [GJ79]. We will review some of the basics of each here.

We assume that the reader is familiar with the classes $\mathbf{L}$ (logarithmic space), $\mathbf{P}$ (polynomial time), $\mathbf{NP}$ (nondeterministic polynomial time), $\mathbf{coNP}$, $\mathbf{\Pi_2^p}$ ($= \mathbf{coNP^{NP}}$), and $\mathbf{\Sigma_2^p}$ ($= \mathbf{NP^{NP}}$), and the poynomial (time) hierarchy $\mathbf{PH} = \bigcup_i \mathbf{\Sigma_i^p}$. Our completeness proofs use the standard complete problem for $\mathbf{\Pi_2^p}$: deciding whether a formula of the form $(\forall x_1, \ldots, x_n)(\exists y_1, \ldots, y_m)\varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is true.

Most of the reductions we consider in this paper are m-reductions (Karp-reductions). We say A *m-reduces* (or *many-one reduces*) to B if there is a function f such that for all x, $x \in A$ if and only if $f(x) \in B$. For most of the paper it will be understood that f is computable in polynomial time. In Section 8, however, we need logarithmic space reductions, because we are classifying problems within $\mathbf{P}$. In one instance will we see another kind of reduction: a *truth-table reduction* from A to B is a function f, computable in polynomial time, where $f(x)$ codes a Boolean formula with atoms of the form $y \in B$, such that $x \in A$ if and only if $f(x)$ is true if interpreted over B.

In this paper graphs are finite and simple (no loops or multiple edges). For a graph $G = (V, E)$ the *order of* G is $|G| := |V|$. The *size of* G is $|E|$. We will use uppercase roman letters to denote graphs. Several letters are reserved for special graphs:

K_n**,** the complete graph on n vertices,

$K_{n,m}$**,** the complete bipartite graph on n and m vertices,

P_n**,** the path on n vertices (hence of length $n-1$), and

C_n**,** the cycle on n vertices.

The graph $K_{1,n}$ is called a *star*, made up of a *center* vertex (of degree n) and n *outer* vertices. We can multiply numbers with graphs: nG is a graph consisting of n copies of G.

A subgraph H of G is called *induced* if H is the result of restricting G to the vertices of H. By a *coloring* of a graph F we mean an edge-coloring of F, that is an assignment of a color to each edge of F. In this paper we will only consider colorings with two colors: red and blue.

Definition 2.1 $F \to (G, H)$ *if every coloring of* F *with red and blue contains either a red subgraph* G*, or a blue subgraph* H*.*

We call a graph F *Ramsey-minimal*, if $F \to (G, H)$, but no proper subgraph F' of F fulfills $F' \to (G, H)$.

The definition of induced arrowing is similar, but we have to be a bit careful.

Definition 2.2 $F \rightarrowtail (G, H)$ *if every coloring of* F *with red and blue contains an induced red subgraph* G*, or an induced blue subgraph* H*. Note that* G *and* H *have to be induced subgraphs of* F*, not just of its red or blue components.*

3 The Second Level of PH

Compared to its downstairs neighbors **NP** and **coNP**, the second level of the polynomial hierarchy is not rich in natural problems. However, in the twenty years since the hierarchy was first defined, some problems have been placed there, and several of them have been shown to be complete.

It is a widely accepted view that a complexity class is justified by its complete problems. Both the number of problems and their naturalness play a role. The classes **P** and **NP** excel on both accounts and have become the most popular classes of computational complexity even outside the field. Nothing similar is true for the higher levels of the polynomial hierarchy. Garey and Johnson [GJ79, Section 7.2] went so far as to say that the interest of the hierarchy was mainly theoretical and not practical: would we really care if the hierarchy collapsed to the second level as long as $\mathbf{P} \neq \mathbf{NP}$? There is some evidence, however, that even $\mathbf{\Sigma_2^p}$ and $\mathbf{\Pi_2^p}$ are natural classes of more than theoretical interest, and the rest of the introduction is a short survey of some of this evidence. There are several results in logic (non-classical logics mostly) which we will ignore in the following discussion.

Probably the first natural problem to be shown $\mathbf{\Sigma_2^p}$-complete was INTEGER EXPRESSION INEQUIVALENCE by Stockmeyer in the early seventies shortly after the polynomial hierarchy was defined.[1] A couple of problems related to INTEGER EXPRESSION INEQUIVALENCE were later shown to be $\mathbf{\Pi_2^p}$-complete and $\mathbf{\Sigma_2^p}$-complete by Wagner [Wag86]. (Wagner even mentions a $\mathbf{\Sigma_3^p}$-complete problem.)

INTEGER EXPRESSION INEQUIVALENCE, roughly speaking, asks whether two representations denote the same object. There are at least two other completeness results for the second level sharing this basic structure. Sagiv and Yannakakis [SY80] proved that deciding whether two monotonic relational expressions are equivalent is $\mathbf{\Pi_2^p}$-complete (monotonic expressions contain only the operators select, project, join and union), and Huynh [Huy84] shows that deciding whether two context-free grammars with only one terminal letter generate the same language is $\mathbf{\Pi_2^p}$-complete. In a similar spirit Lin [Lin95] recently showed a problem related to pattern matching (and program optimization) to be $\mathbf{\Sigma_2^p}$-complete.

Dependencies in a concurrent system can be described and analyzed using POMSETS, which are labeled, acyclic, directed graphs. The edges in the graph restrict the order of events (vertices). A word on the set of labels (actions) is *consistent*, if there is an allowed order of events that produces it. This associates a language $L(P)$ (of consistent words) with each POMSET P. Feigenbaum, Kahn and Lund showed that POMSET LANGUAGE CONTAINMENT, the problem of deciding whether $L(P) \subseteq L(Q)$ is $\mathbf{\Pi_2^p}$-complete [JF93].

Ko and Tzeng [KT91] exhibited three $\mathbf{\Sigma_2^p}$-complete problems that belong to the realm of computational learning theory. The problems are: PATTERN CONSISTENCY which asks whether for two sets

[1] The completeness result appears both in a conference paper of 1973 authored by both Stockmeyer and Albert Meyer, and in a 1976 paper by Stockmeyer on the polynomial hierarchy [GJ79, Problem AN18].

of words there is a pattern that matches each word in one of the sets but no word in the other, GRAPH RECONSTRUCTION asking whether for two sets of graphs there is a graph G such that all graphs in the first set are isomorphic to a subgraph of G but none of the graphs in the second set is, and, finally, GENERALIZED 3-CNF CONSISTENCY which asks whether for two sets of Boolean formulas there is a 3-CNF formula which is consistent with each formula from the first set but with no formula from the second set. Ko and Tzeng isolate a pattern common to these three problems, but they also show that this pattern comprises problems not $\mathbf{\Sigma_2^p}$-complete. It would be interesting to find a generalization of the three problems that does imply $\mathbf{\Sigma_2^p}$-completeness.

Some famous problems remain candidates for $\mathbf{\Sigma_2^p}$-completeness. In the sixties and seventies the question of size of machines and programs was investigated in detail, and in parallel to research done in computability, Meyer and Stockmeyer defined the MINIMAL problem of deciding whether for a given formula φ there is no formula of smaller size that is equivalent to it. Later Stockmeyer defined the refined problem $\mathsf{MEE}_{\mathsf{DNF}}$ of deciding whether for a given DNF formula φ there is a DNF formula of size at most k which is equivalent to it. Garey and Johnson [GJ79] also mention the variant MEE where the requirement that φ be in DNF is dropped. Both $\mathsf{MEE}_{\mathsf{DNF}}$ and MEE are in $\mathbf{\Sigma_2^p}$ whereas MINIMAL is in $\mathbf{\Pi_2^p}$. It is a trivial observation that $\mathsf{MEE}_{\mathsf{DNF}}$ and therefore MEE are $\mathbf{\Pi_1^p}$-hard. Only recently did Hemaspaandra and Wechsung [HW97] show that MINIMAL is also $\mathbf{\Pi_1^p}$-hard. For MEE and $\mathsf{MEE}_{\mathsf{DNF}}$ they pushed up the lower bound to $\mathbf{P}_{||}^{\mathbf{NP}}$ (nonadaptive queries to $\mathbf{NP}$). Recently Umans [Uma98] showed that $\mathsf{MEE}_{\mathsf{DNF}}$ is indeed $\mathbf{\Sigma_2^p}$-complete. (Even approximating the size of a minimal equivalent DNF formula remains $\mathbf{\Sigma_2^p}$-complete [Uma99].) The other variants remain open (although it is possible to construct artificial size measures for which all of these problems are complete).

Finally, there is an analogue of the GRAPH ISOMORPHISM problem, the BOOLEAN ISOMORPHISM problem in $\mathbf{\Pi_2^p}$ which asks whether two formulas are equivalent up to a permutation of the variables. BOOLEAN ISOMORPHISM is $\mathbf{\Pi_1^p}$-hard but not known to lie in $\mathbf{\Pi_1^p}$. Agrawal and Thierauf [AT96] recently showed that the complement of BOOLEAN ISOMORPHISM lies in $\mathbf{AM}^{\mathbf{NP}}$ which means that we do not expect it to be complete for $\mathbf{\Sigma_2^p}$ since this would collapse the polynomial hierarchy to $\mathbf{\Sigma_3^p}$.

4 Arrowing

Theorem 4.1 ARROWING *is* $\mathbf{\Pi_2^p}$*-complete.*

This result will be immediate from Lemma 4.11 which shows that deciding $F \to (K_{1,p}, K_n)$ for any fixed $p \geq 2$ is $\mathbf{\Pi_2^p}$-complete. We will eventually strengthen this result to show that deciding $F \to (T, K_n)$ is $\mathbf{\Pi_2^p}$-complete for any tree T of size at least two, but so as not to obscure the basic construction we start with the special case. Some intermediary results will be proved in full generality, however, for later use.

Definition 4.2 *A coloring of F is called* (G,H)-good *if F does not contain a red G or a blue H in this coloring.*

In all of the following constructions we will use the implicit convention that if we take several graphs and identify some of their edges or vertices, then only these edges or vertices are identical and the remainders of the graphs still distinct.

Definition 4.3 *A graph F is called a* (G,H)-enforcer with signal vertex v *if the graph obtained from F by attaching a new edge to v has the property that this edge is colored blue in all (G,H)-good colorings, and there is a (G,H)-good coloring of the graph.*

We can use enforcers to force edges leaving the signal vertex to be blue (assuming that H is 2-connected). We will sometimes talk of a blue enforcer to make this point explicit. Red enforcers are defined analogously.

Enforcers will be essential in the completeness proof. Let us see how to construct them for the special case of trees. For this we need a well-known fact from the Ramsey theory of graphs. The *Ramsey number* $r(G,H)$ of two graphs is defined as the least n such that $K_n \rightarrow (G,H)$.

Lemma 4.4 (Chvátal [Chv77]) *If T is a tree on t vertices, then $r(T,K_n) = (t-1)(n-1)+1$.*

The lower bound of Chvátal's result is established by coloring a subgraph $(n-1)K_{t-1}$ of $K_{(t-1)(n-1)}$ red and its complement blue. Note that in this coloring the red degree of each vertex is $t-2$. This observation can be generalized.

Lemma 4.5 *The red degree of every vertex in $K_{(t-1)(n-1)}$ in any (T,K_n)-good coloring is at least $t-2$, where t is the order of T.*

Proof. Let $m = (t-1)(n-1)$. Assume for a contradiction that there is a (T,K_n)-good coloring of K_m in which some vertex x has red degree $d \leq t-3$. Let $y_1,\ldots,y_d$ be its red neighbors. Look at the graph $K_m - \{x,y_1,\ldots,y_d\}$ under the induced coloring. This graph has $m-(d+1) \geq m-(t-2) = (t-1)(n-1)-(t-2) = (t-1)(n-2)+1$ vertices and hence (by Chvátal's result) contains a red T or a blue K_{n-1}. Neither, however, is possible, since a red T would also be contained in the original graph, and a blue K_{n-1} together with x would form a blue K_n in the original graph. $\diamondsuit$

It is a folklore fact in graph theory (a proof can be found in Chvátal's note [Chv77]) that a graph of minimum degree $t-1$ contains every tree on t points. More precisely the tree can be embedded node by node, starting at an arbitrary node of the tree and the graph. Hence if we remove from T an edge leading to a leaf, then the resulting tree T' will occur as a red subgraph in any (T,K_n)-good coloring of $K_{(t-1)(n-1)}$ with any particular vertex of T' in any particular vertex of $K_{(t-1)(n-1)}$. Hence $K_{(t-1)(n-1)}$ is an enforcer, and each of its vertices can act as a signal vertex for $n>2$. (While the previous lemma is still true for $n=2$, it is no longer possible to find a good coloring in this case.)

Lemma 4.6 *For any tree T a (T,K_n)-enforcer can be constructed in polynomial time in $n>2$ (in unary) (namely take a $K_{(|T|-1)(n-1)}$ and any of its vertices).*

Before we can prove Theorem 4.1 we need to construct a particular kind of graph that works like a switch between two edges: exactly one of the edges will be blue in a good coloring, and either can be blue.

Definition 4.7 *A graph F is called a (G,H)-*switch *if it contains two edges e and f such that for all (G,H)-good colorings exactly one of e and f is red while the other is blue. Furthermore there has to be a (G,H)-good coloring of F where e is red and f is blue, and a (G,H)-coloring where e is blue and f is red.*

Our switches are called *negative senders* in the combinatorial literature. We prefer the term switches in this paper since it better reflects the purpose of the graph.

The construction of switches requires yet another kind of graph which like the enforcer determines the color of a particular edge.

Definition 4.8 *A graph is called a (G,H)-*determiner with signal edge e *if e is colored red in all (G,H)-good colorings, and there is a (G,H)-good coloring of the graph.*

For later reference note that we will sometimes call these determiners red determiners to distinguish them from blue determiners (which are defined analogously).

Lemma 4.9 *Suppose that G is a connected graph of size at least two and that we can compute a (G, K_n)-enforcer in polynomial time in n (in unary). Then a (G, K_n)-determiner can be constructed in polynomial time.*

Proof. Build a graph F from a copy of K_n and $n-2$ copies of a (G, K_n)-enforcer as follows: label two vertices of the K_n with u and v, and identify every other vertex in K_n with the signal vertex of a (G, K_n)-enforcer. We claim that F is a (G, K_n)-determiner. Fix a (G, K_n)-good coloring of F. The enforcers force all edges of K_n except for uv to be blue. This forces uv to be red since the coloring does not contain a blue K_n. Furthermore, coloring all the enforcers with a good coloring and letting all the edges in K_n be blue except for uv, which is red, shows that there is a good coloring of F. Note that this relies on the fact that G is connected. $\diamondsuit$

The construction of switches for connected graphs in general seems to be hard, the problem being that connected graphs that are not 2-connected (like trees) can cause trouble when several coding devices are merged. More precisely, the graphs might extend through several devices making a good coloring impossible. Hence we will first concentrate on the easier case of stars ($K_{1,p}$), and only later show how to generalize the construction.

Lemma 4.10 *A $(K_{1,p}, K_n)$-switch can be constructed in polynomial time (for fixed $p \geq 2$). The switch has the additional property that if we attach additional edges in vertices incident with a signal edge of the switch, these new edges are forced to be blue in any good coloring.*

Proof. Fix p. Construct a graph F from $K_{1,p-2}$ by taking a red $(K_{1,p}, K_n)$-determiner for each edge of the $K_{1,p-2}$ and identifying that edge with the signal edge of the determiner. (If $p = 2$, then $K_{1,p-2}$ is just an isolated vertex.) Let v be the center vertex of the $K_{1,p-2}$. If we attach new edges to v in F, then in any $(K_{1,p}, K_n)$-good coloring of F at most one of these edges can be red. Furthermore, there is a good coloring with any of these edges being red, and all the others blue. Let us call F a weak switch: it forces the number of red edges out of its signal vertex v to be at most one.

With the weak switch we can now construct the switch. Take three copies of a K_n, and let u_i, v_i, w_i be a triangle in the ith copy ($1 \leq i \leq 3$). Identify $v_1 = u_2$, $w_2 = v_3$, and $w_1 = u_3$. (See Figure 2; the figure only shows the triangle parts of the K_n.)

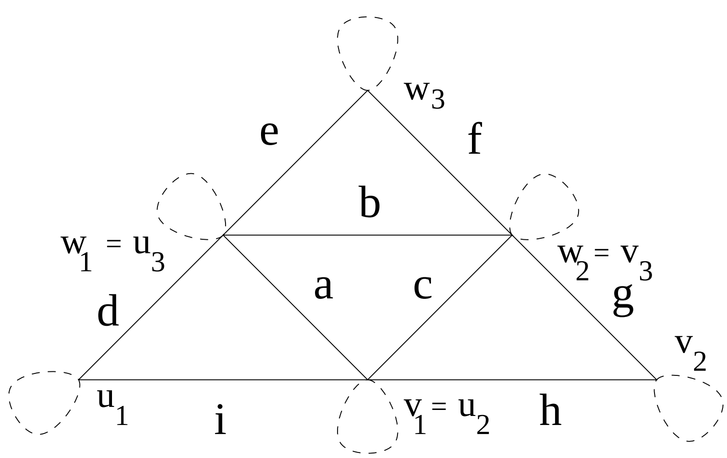

Figure 2: A $(K_{1,p}, K_n)$-switch

Furthermore, make each vertex in the resulting graph, with the exception of the labeled ones, the signal vertex of a $(K_{1,p}, K_n)$-enforcer. Finally identify each of u_1, $v_1 = u_2$, v_2, $w_2 = v_3$, w_3, and $w_1 = u_3$

with the signal vertex of a weak switch. Let $e = u_3w_3$, and $f = v_3w_3$. We claim that the graph we constructed with signal edges e and f fulfills the definition of a switch.

To this end we remark that in a good coloring it is true that:

(i) Each triangle u_i, v_i, w_i contains a red edge (otherwise it would complete a blue K_n);

(ii) a, b, and c are blue (assume for example b was red, then d, a, c and g are forced to be blue by the weak switches in w_1 and w_2; Hence both i and h are forced to be red by (i), forcing the weak switch in v_1 to fail);

(iii) The edges d, e, f, g, h, i are alternatingly colored red and blue in this order with d either red or blue (by (i), (ii), and the properties of the weak switches).

Furthermore, e can be either red or blue in a good coloring. For the verification of this last claim we combine good colorings of the devices that were used in the construction of the switch. Note that the devices are merged by identifying at most one vertex, or at most one edge, in two different devices. In both cases, parts of different components cannot add up to a blue K_n since it is 3-connected for $n \geq 4$. It is also easy to check that we do not add any red stars when we build the weak switches from determiners, or attach the weak determiners to the triangles. Hence we get a good coloring of the devices, which we extend by coloring a, b, and c blue, and the other edges alternatingly red and blue (starting with d being either red or blue).

We note that additional edges out of the endpoints of e or f will be forced to be blue in any good coloring, because all of the signal vertices of the weak switches attached to u_3, v_3, and w_3 are incident with one red edge (e, f, g or d).

Finally, we should observe that the construction can be performed in polynomial time. $\diamondsuit$

With this special switch we can complete the construction.

Lemma 4.11 *Let $p \geq 2$. For any Π_2 sentence ψ we can construct a graph F and compute a number n in polynomial time such that ψ is true if and only if $F \to (K_{1,p}, K_n)$.*

Proof. Let $\psi = (\forall x_1, \ldots, x_k)(\exists y_1, \ldots, y_k)[\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)]$ be the formula to be coded with its $2k$ variables ranging over $\{0, 1\}$ with φ in conjunctive normal form. We will refer to the x_i as the *x-variables* and the y_i as the *y-variables.*

If we have a formula $\rho(x, \overline{x})$ which has occurrences of a variable x both positively and negatively, then the following is true: $(\forall x)[\rho(x, \overline{x})] \equiv (\forall x)(\exists y)[(\overline{x} \vee y) \wedge (x \vee \overline{y}) \wedge \rho(y, \overline{y}).]$ We apply this equivalence to φ, and can therefore assume that each x-variable occurs only twice, once positively and once negatively with one y variable each (and no other variables). Applying the usual transformations allows us to assume that φ contains at most three literals per clause and each literal occurs at most twice in the whole formula. We can also assume that each variable occurs at most once per clause. Let $\varphi(a, b)$ consist of m clauses $C_1, \ldots, C_m$.

We will now mimic a reduction from SAT to CLIQUE (see for example [Pap94, Theorem 9.4]).

Construct the graph F as follows. For each clause C that does not contain an x-variable take as many vertices as there are literals in the clause, and label the vertices by the literals in the clause. Call this a y-gadget. For each pair of clauses $(\overline{x}_i \vee y_j)$ and $(x_i \vee \overline{y}_j)$ take a $(K_{1,p}, K_m)$-switch with adjacent signal edges e_i and f_i. Since these two edges are different there is a vertex u of e_i which does not belong to f_i and a vertex v of f_i which does not belong to e_i. Label u with y_j and v with $\overline{y}_j$ and the common vertex of e_i and f_i with x_i. Call this an x-gadget.

Include all edges between labeled vertices that belong to different gadgets and are not labeled by contradictory literals. Furthermore make each vertex of a y-gadget the signal vertex of a $(K_{1,p}, K_m)$-enforcer.

This completes the construction of F (for an example see Figure 3). We claim that $F \to (K_{1,p}, K_m)$ if and only if ψ is true.

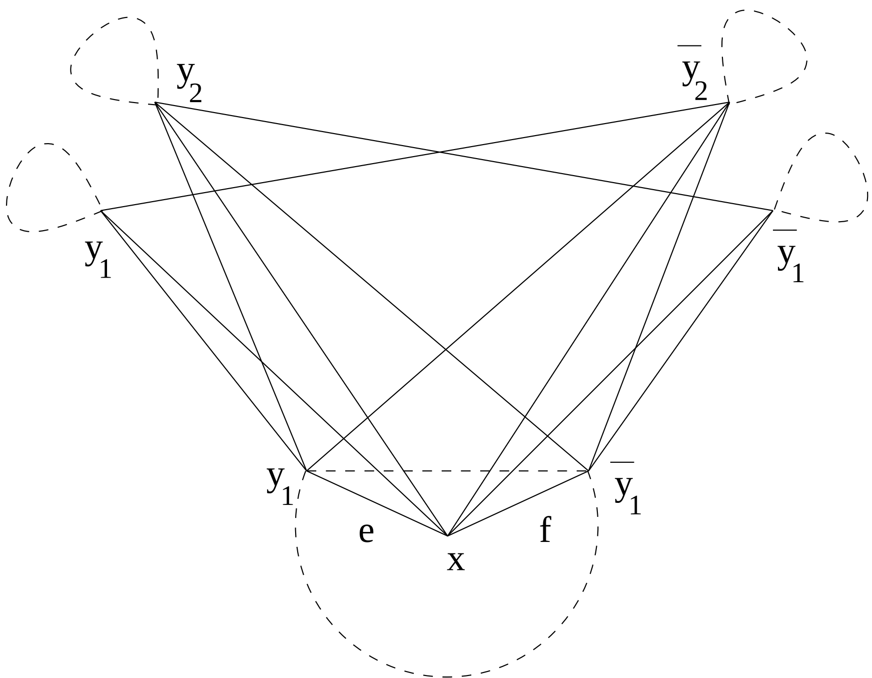

Figure 3: F for $(\forall x)(\exists y)[x \not\leftrightarrow y] \equiv (\forall x)(\exists y_1, y_2)[(x \vee \overline{y}_1) \wedge (\overline{x} \vee y_1) \wedge (y_1 \vee y_2) \wedge (\overline{y}_1 \vee \overline{y}_2)]$

Assume first that ψ is true. Suppose for a contradiction that there is a $(K_{1,p}, K_m)$-good coloring of F. Fix that coloring. In this coloring all the enforcers and switches work, namely all the edges between the gadgets are blue. The edges between a y-gadget and another gadget are blue because of the enforcers in the y-gadget. The edges between different x-gadgets are blue because of the special nature of the switch we constructed (see Lemma 4.10). Furthermore, within each x_i-gadget at least one of e_i and f_i is blue. (Indeed exactly one is blue, but we do not need that for the proof.) Call x_i true if e_i is blue, and false otherwise. For this truth assignment to the x-variables there is a truth assignment to the y variables such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is true. Fix this assignment. For each y-gadget pick a vertex whose label is assigned true. In each x-gadget choose the two vertices belonging to e_i if x_i is true and f_i otherwise. By the construction of F these m vertices form a blue clique contradicting the assumption.

To show the other direction assume that ψ is false, that is, there is an assignment of truth-values to $x_1, \ldots, x_k$ such that for all assignments of truth-values to $y_1, \ldots, y_k$ the formula $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is false. Fix such $x_1, \ldots, x_k$. The enforcers and switches were constructed in such a way that they have good colorings. Fix a good coloring for each enforcer. For the x_i-gadget we choose a coloring of the switch in which e_i is blue and f_i is red if x_i is true and vice versa otherwise. Finally, color all edges between gadgets blue. We claim that this coloring is a $(K_{1,p}, K_m)$-good coloring. It obviously does not contain a red $K_{1,p}$ since none of the components contains a red $K_{1,p}$, and all the edges between gadgets are blue (here we use that $K_{1,p}$ is connected). Suppose, for a contradiction, that there is a blue K_m.

Each y-gadget can share at most one vertex with this K_m and each x-gadget at most two. Hence every y-gadget has exactly one vertex in common with K_m and every x_i-gadget exactly two. The two vertices in the x_i gadget could be y_j and $\overline{y}_j$ (there is a blue edge between them), but since the set of vertices reached from x_i by a blue edge is a superset of the set of vertices reached from either y_j or $\overline{y}_j$ by a blue edge, and one of e_i, f_i is blue, we can assume that one of the two vertices is x_i. Call a y-literal true if one of the K_m vertices is labeled with that literal. By the construction (and the preceding remark) this yields a well-defined partial assignment of truth-values to y-variables such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)$ is true, contradicting the choice of the assignment to the x variables. $\diamondsuit$

We will now generalize the result to the case of trees against cliques.

Theorem 4.12 *Deciding $F \to (T, K_n)$ is $\mathbf{\Pi_2^p}$-complete for any fixed tree T of size at least two.*

Proof. We already know how to build (T, K_n)-enforcers, the task left for the proof of the theorem is the construction of a switch which fulfills the extra condition mentioned in Lemma 4.10, that is, additional edges out of the signal vertices must be forced to be blue. We distinguish two cases.

Case 1. T contains a vertex of degree two which has exactly one leaf as a neighbor.

Let v be the other neighbor of that vertex, and T' be the subtree of T which is obtained by removing the leaf and its common neighbor with v. Take two copies T_1 and T_2 of T' and add an edge between v_1 and v_2, the copies of v in T_1 and T_2. The resulting graph T^* contains T as a subtree: take T_1, the edge $v_1 v_2$, and any edge incident with v_2 in T_2 (there has to be such an edge, otherwise v would be a leaf in T).

Construct a new graph G from T' by making each of its edges the signal edge of a red determiner. The resulting graph G has the following property: if we attach a path of length two to v in G, then one of the two edges of that path has to be blue in every good coloring. (G corresponds to the weak switch we used in the case $T = K_{1,p}$.)

Take a copy of a K_n and let u, v, w be three vertices in that copy. Add $n - 3$ copies of a (T, K_n)-enforcer and two copies of G. Identify the signal vertices of the enforcers with the unlabeled vertices, and the signal vertices of the two copies of G with u and v respectively (as in Figure 4).

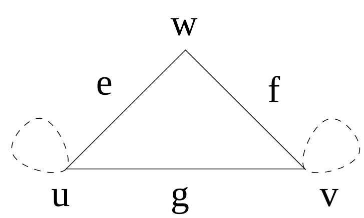

Figure 4: A (T, K_n)-switch (Case 1)

We claim that this graph works as a switch. In any good coloring of this graph:

(*i*) g is colored blue (Since T^* contains T as a subtree, a red g would complete a red T);

(*ii*) Exactly one of e and f is red (at least one because the triangle contains at least one red edge, and at most one because of the copies of G);

(*iii*) Additional edges out of w are forced to be blue (one of the two copies of G forced this through the red edge to w).

Moreover, there are good colorings in which either of e or f is red (merge good colorings of devices, make g blue, and either of e or f red and the other blue). With this switch we can repeat the completeness construction of Lemma 4.11. The only thing to notice is that edges between x-gadgets are forced to be blue in a good coloring. They must be blue between w vertices because of (iii) above, and between u and v vertices because edges between copies of G are forced to be blue (as in (i) above).

Case 2. T contains a vertex that is adjacent to at least two leaves.

Let v be a vertex of T adjacent to at least two leaves, and T' be the subtree of T obtained by removing exactly two leaves adjacent to v. As above let T^* be two copies of T' with their v-vertices connected by an edge. T will not occur as a subtree of T^* (as proved in the next lemma).

Again construct a graph G from T' by making each of its edges the signal edge of a red determiner. The resulting graph G has the following property: if we attach at least two edges to v in G, then at most one of those edges can be red in any good coloring. (We called such a graph a weak switch in the proof of Lemma 4.11.)

We construct the switch as follows:

Take three copies of a K_n, and let u_i, v_i, w_i be a triangle in the ith copy ($1 \leq i \leq 3$). Identify $v_1 = u_2$, $w_2 = v_3$, and $w_1 = u_3$ (as an illustration the earlier Figure 2 will serve). Furthermore, make each vertex in the resulting graph, with the exception of the labeled ones, the signal vertex of a (T, K_n)-enforcer. Finally identify each of u_1, $v_1 = u_2$, v_2, $w_2 = v_3$, w_3, and $w_1 = u_3$ with the signal vertex of a copy of G. Let $e = u_3w_3$, and $f = v_3w_3$. We claim that the graph we constructed with signal edges e and f fulfills the definition of a switch.

In a good coloring:

(i) Each triangle u_i, v_i, w_i contains a red edge (otherwise it would complete a blue K_n);

(ii) a, b, and c are blue (Assume for example that b was red. Then d, a, c and g are forced to be blue by the copies of G in w_1 and w_2. Hence both i and h are forced to be red by (i), contradicting the fact that there is a copy of G in v_1);

(iii) Exactly one of e and f is red (at least one of them is red by (i) and (ii), and at most one of them is red because of the copy of G in w_3), and similarly for d, i and g, h;

(iv) Exactly one edge in each pair $\{e, d\}$ and $\{f, g\}$ is red.

We note that there is a good coloring of the graph in which e is red, and one in which f is red. For this we need the assumption that T is not a subgraph of T^*. This assumption implies that there is a good coloring of two copies of G whose signal vertices are connected by a red edge. Note that the switch will force additional edges attached to u_3, v_3 or w_3 to be blue. (For example, one of e or d is always red in a good coloring, so the copy of G in w_1 will force further edges attached to w_1 to be blue.)

With this new switch the remainder of the construction is the same as in Lemma 4.11. ◇

For the following lemma let T, T', and T^* be as in the proof above, namely T' has a vertex v such that T is isomorphic to T' with two edges attached to v, and T^* is isomorphic to two copies of T' whose v-vertices are connected by an edge.

Lemma 4.13 *T does not occur as a subgraph of T^*.*

Proof. Let T be a smallest counterexample to the claim of the lemma. Assume that T' has n edges $\{e_1, \ldots, e_n\}$. Let T_1 and T_2 be the two copies of T' in T^* and $e_{i,j}$ be the edges of T_i such that $e_{1,j}$ and $e_{2,j}$ correspond to e_j in isomorphisms from T_1 and T_2 to T', respectively. Fix an embedding of T into T^*. Suppose there is a j such that neither $e_{1,j}$ nor $e_{2,j}$ are part of the embedding of T. Remove the edge e_j from T' splitting it into two components. Let S' be the component which contains the vertex

v of T'. From S' construct S by attaching two new edges to the copy of v in S'. Then S^* contains T as a subgraph, and therefore S itself (since S is a subgraph of T). Hence S would be a smaller counterexample than T, contradicting the choice of T.

Consider the embedding of the $n + 2$ edges of T in T^*. One edge of T is taken care of by the edge connecting T_1 and T_2. That leaves $n + 1$ edges. Since at least one edge from each of the n pairs $\{e_{1,j}, e_{2,j}\}$ has to belong to the embedding it follows that for $n - 1$ pairs exactly one edge belongs to the embedding of T, and for one pair both edges belong to the embedding. Without loss of generality that one pair is $e_{1,1}$ and $e_{2,1}$. Now $e_{1,1}$ has to be incident with v_1 (the vertex corresponding to v in T_1), since otherwise all the edges between $e_{1,1}$ and v_1 and the edges between $e_{2,1}$ and v_2 would be part of the embedding of T. This would contradict the fact that there is only one pair for which both edges are part of the embedding. Let A be the component of T_1 that is part of the embedding of T and is separated from v_1 by $e_{1,1}$. Let C be the remaining edges in T_1 belonging to the embedding of T. Similarly define B and D as subsets of edges in T_2. See Figure 5 for an illustration.

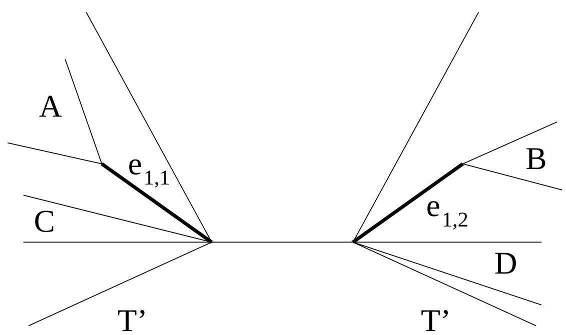

Figure 5: Partitioning the embedding of T

Via the isomorphisms we can look at all these sets as subsets of the edges of T' and we note that A, B, C, D and e_1 form a partition of T' into subtrees. This gives us a new way to look at T illustrated on the right side in Figure 6. On the left side is the view of T as embedded in T^* (from the previous figure).

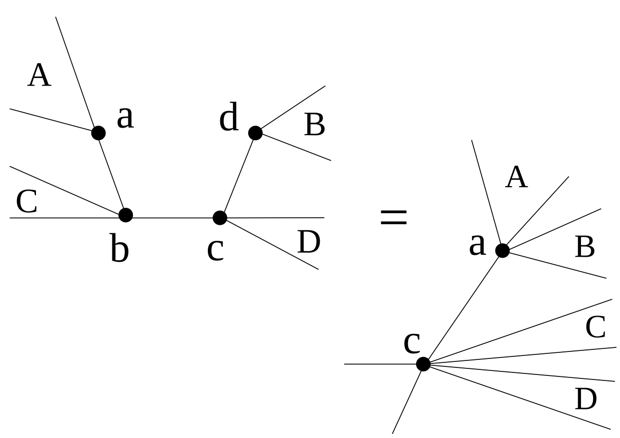

Figure 6: Two views of T

We conclude the proof by counting the number of leaves in each representation of T. We consider

A, B, C, and D as rooted subtrees of T with roots a, b, c, and d respectively. For these four sets let $l(A)$, $l(B)$, $l(C)$, and $l(D)$ denote the number of leaves in A, B, C, and D respectively. We do not count the root as a leaf.

The left drawing of T shows that T has $\max\{l(A), 1)\} + \max\{l(B), 1\} + l(C) + l(D)$ leaves, in the right drawing, however T has $2 + \max\{l(A) + l(B), 1\} + l(C) + l(D)$ leaves. It is easy to see that these two numbers cannot be equal. $\diamondsuit$

Stronger versions of the lemma can be found in [SS99].

5 Strong Arrowing

Theorem 5.1 STRONG ARROWING *is* $\mathbf{\Pi_2^p}$*-complete.*

Determining the complexity of strong arrowing can be done using the same approach as in the non-induced case. The result, however, will be weaker: we are only able to show the $\mathbf{\Pi_2^p}$-completeness of $F \rightarrowtail (K_{1,p}, K_n)$. The problem is that the combinatorics are more involved. At the heart of the completeness proof for ARROWING was the construction of enforcers, building on the computation of $r(T, K_n)$ by Chvátal. Exact *induced* Ramsey numbers are only known for small graphs. Even finding any F with $F \rightarrowtail (G, H)$ for a given G and H is hard. The existence of such an F for every pair (G, H) was only shown in 1973, independently, by Deuber, by Erdős, Hajnal, and Pósa, and by Rödl [Die97]. Though two of these proofs are constructive, the graph F they construct will be at least of doubly exponential size in the input graphs. Since we are only allowed polynomial time we cannot use them. Though there are better bounds for induced Ramsey numbers in the literature (see for example the recent paper by Kohayakawa, Prömel, and Rödl [KPR98]) these bounds are obtained by the use of random graphs, and therefore of no help in polynomial-time constructions.

We are able to solve the special case of stars versus cliques, and we give a further example in which we show how to determine the complexity of $F \rightarrowtail (P_4, K_n)$. (Recall that P_4 is a path on four vertices.)

Since we are now dealing with induced graphs we redefine the basic notion of a good coloring for the rest of this section.

Convention. In this section a coloring of a graph F is called (G, H)*-good* if it contains neither a red G nor a blue H as an induced subgraph of F.

This, of course, changes the notions of enforcers and determiners as defined above, and to avoid confusion we will call the gadgets we construct for the induced case strong enforcers and strong determiners.

The goal of this section is to show to what extent the ideas of the completeness proof for ARROWING work for the induced case. The main observation is that the proof of Lemma 4.11 goes through for the induced case as well. The only changes necessary are the use of strong enforcers instead of enforcers and a modification to the construction of the switch.

Lemma 5.2 *If we can build strong* (T, K_n)*-enforcers in polynomial time (in* T *and* K_n*), then deciding* $F \rightarrowtail (T, K_n)$ *is* $\mathbf{\Pi_2^p}$*-complete for any fixed tree* T *of size at least two.*

Proof. We will follow the proof of Theorem 4.12, pointing out the necessary changes.

Let G be the strong (T, K_n)-enforcer with signal vertex v from the assumption of the lemma (we assume $n > 2$). We need the enforcer to have some additional properties for the proof. We will achieve this by rebuilding it. In Lemma 4.9 we showed how to construct a (H, K_n)-determiner from a (H, K_n)-enforcer. The same construction yields a strong (T, K_n)-determiner if we start with the strong (T, K_n)-enforcer G.

Let T' be isomorphic to T with one leaf (and its corresponding edge) removed. Let u be the vertex in T' that was incident (in T) with the removed edge. From T' we construct the strong (T, K_n)-enforcer G' by making each edge of T' the signal edge of a (new) (T, K_n)-determiner. Then u in G' is the signal vertex of a strong (T, K_n)-enforcer. Namely if we attach a new vertex v to u via edge e, then e is forced to be blue in all good colorings of $G' \cup \{e\}$, and there is a good coloring of that graph. Consider what happens to this graph we add edges from v to all neighbors of u in T' (as a subgraph of G'). In a good coloring of that graph e is no longer forced to be blue. We claim there is a good coloring in which e is red, and there is a good coloring in which e is blue. For these colorings fix a good coloring of G'. Color all edges from v to neighbors of u in T' blue, and e blue or red, as required. We claim that the resulting colorings are good. The reason is that u and v cannot both be part of an induced red subtree, since they are connected by at least one common neighbor, forming a triangle. Furthermore, we are not creating any new blue cliques with the blue edges we are adding, since none of the neighbors of u in T' are adjacent (since T' is a tree).

Figure 7 shows as an example the case $T = K_{1,2}$. In this case T' consists of a single edge f which is forced by a determiner to be red. Hence e is forced to be blue, but e', on the other hand, is not. It can be either red or blue in a good coloring. We say that the determiner is *disabled* for e'.

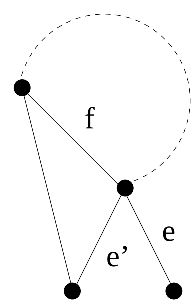

Figure 7: G' forcing e, and disabled for e' where $T = K_{1,2}$

In this fashion we can disable the enforcer G' for any number of edges out of the signal vertex. Note that the construction will still work if we add two strong enforcers G', one to each end of an edge $e = uv$, and disable both of them on edge e. The additional edges from the strong enforcer attached to v to the vertex u are colored blue in the good coloring described above, and since there are no other connections between the enforcers, these edges do not complete a new blue clique in the enforcer in u.

We can now construct a weak (T, K_n)-switch as follows. Take a copy of T and let $e = uv$ and $f = vw$ be two adjacent edges in T. Make each of u, v, and w a signal vertex of a copy of G', and disable the forcing of all edges in T (with regard to any of these three enforcers). Finally, make all edges of T with the exception of e and f signal edges of (T, K_n)-determiners. This completes the weak switch. In a good coloring of the weak switch all edges of T, with the exception of e and f, are forced to be red. Hence either e or f has to be blue (otherwise they would complete a red induced T). Furthermore, there is a good coloring in which e is red and there is a good coloring in which f is red. We obtain these by combining good colorings of the components. Also note that the weak switch fulfills the requirement that all additional edges out of u, v, and w are forced to be blue.

With the weak switch we can now repeat the construction from Lemma 4.11. In the non-induced case we constructed a switch from the weak switch. The reason for this was not that the proof required that exactly one of e and f is blue, but instead that we needed to force the color of additional outgoing edges. The completeness proof remains unchanged. This is because the only place where we argue with the absence of a red T (a red induced T, respectively) is in guaranteeing that the gadgets work as expected. The remainder of the construction and the proof only works with blue cliques for which non-induced and induced have the same meaning. $\diamondsuit$

We can now apply the theorem.

Corollary 5.3 *Deciding whether* $F \rightarrowtail (K_{1,p}, K_n)$ *is* $\mathbf{\Pi_2^p}$*-complete.*

The proof of the corollary is completed with the construction of a strong $(K_{1,p}, K_n)$-enforcer in the next lemma. As in the non-induced case there is a connection to the construction of graphs F for which $F \rightarrowtail (K_{1,p}, K_n)$. We already mentioned above the bounds on the size of such an F given by the induced Ramsey theorem are not good enough and the polynomial bounds in the literature are achieved by using random graphs. Hence we have to prove our own induced subgraph result. The paper *Induced Graph Ramsey Theory* [SS00] collects several such constructive results. For the special case at hand we use a construction based on an idea of Thomas Hayes [Hay]. The idea is to use a complete n-partite graph with ip vertices in the ith partition class.

Lemma 5.4 *We can construct a strong* $(K_{1,p}, K_n)$*-enforcer in polynomial time in* p *and* n *(in unary).*

Proof. We will show how to construct F such that $F \not\rightarrowtail (K_{1,p}, K_n)$, and F has a vertex v such that in any $(K_{1,p}, K_n)$-good coloring v is at the center of a red induced $K_{1,p-1}$. Let $F = K_{1,p-1,2p,3p,\ldots,(n-1)p}$, a complete n-partite graph where the ith partition class contains $\max\{1, (i-1)p\}$ vertices, with the exception of the second partition which contains only $p-1$ vertices. Let v be the unique vertex in the first partition. Coloring the $p-1$ edges between the first and second partition red and all the other edges blue shows that $F \not\rightarrowtail (K_{1,p}, K_n)$. Fix any $(K_{1,p}, K_n)$-good coloring of F. Suppose there is a blue edge from v to a vertex in the second partition. Using the fact that there is no red $K_{1,p}$ one easily shows by induction that the first i partition classes contain a blue K_i, finishing the proof. $\diamondsuit$

How can we build enforcers for graphs other than stars? The constructions of determiners and senders in the literature rely on Ramsey-minimal graphs [BNR85]. Remember that a graph is *Ramsey-minimal* for (G, H) if it arrows (G, H) but no proper subgraph does. Another look at the (T, K_n)-enforcer and the strong $(K_{1,p}, K_n)$-enforcer will convince the reader that these constructions contain a trace of the minimality idea. The question is whether this idea can be used more generally. There are two problems with this approach. The first is that we do not know how to find Ramsey-minimal graphs, and the second that there is no known way to turn them into enforcers or senders without an exponential increase in size.

Randomized reductions might help solve the first problem. Most of the randomized constructions in [KPR98] only depend on a polynomial amount of randomness. Hence a randomized polynomial-time algorithm can find F such that $F \rightarrowtail (G, H)$ for a large class of graphs G and H. Then a randomized truth-table reduction can easily find a Ramsey-minimal F for G and H. At this point, however, we do not know how to solve the second problem, even allowing randomness.

We will sketch one example in which we solve these two problems by using a weakened version of Ramsey-minimal graphs. Our modest goal is to show that $F \rightarrowtail (P_4, K_n)$ is $\mathbf{\Pi_2^p}$-complete. We do not quite achieve this goal. The best we can do at present is to show completeness for $\mathbf{\Pi_2^p}$ under truth-table reductions; this is the price we have to pay for our ignorance of Ramsey-minimal graphs. A *truth-table reduction* is a reduction that works in three steps: it first determines which queries it wants to make, then it makes the queries, and finally decides whether to accept or not (without making any further queries).

We can in polynomial time compute a graph F such that $F \rightarrowtail (P_4, K_n)$ (for a proof see [SS00]). Fix an ordering of the edges of F and remove the edges in order one by one until finding a graph F' for which $F' \not\rightarrowtail (P_4, K_n)$. (We use the oracle to do this.) The search terminates since $\emptyset \not\rightarrowtail (P_4, K_n)$. Let the last edge that was removed be between vertices v and w. That is, $F' \cup \{\{v, w\}\} \rightarrowtail (P_4, K_n)$ and $F' \not\rightarrowtail (P_4, K_n)$.

Consider any good coloring of F'. If we add a red edge vw to F' we obtain a graph $F' \cup \{\{v, w\}\}$ which either contains a red induced P_4 or a blue K_n (because vw was the last edge to be removed in the construction of F'). Since the coloring of F' did not contain a blue K_n, and the red vw cannot complete a blue K_n, the edge vw must be part of a red P_4 in $F' \cup \{\{v, w\}\}$. Hence in any good coloring of F' either v and w are both incident with a red edge, or one of them is incident with a red induced P_3. We distinguish two cases:

(i) There is a good coloring of F' in which neither v nor w is incident to the endpoint of a red induced P_3, and hence they are both incident to a red edge in this good coloring; or

(ii) At least one of v or w is incident to the endpoint of a red induced P_3 in every good coloring.

In the first case we take two copies F_1 and F_2 of F' and identify $v := v_1 = v_2$ and $w := w_1 = w_2$ to get a new graph G. We can obtain a good coloring of G by choosing for each F_i the coloring of F' described in case (i). Consider any good coloring of G. We claim that both v and w are incident to a red edge. Suppose for a contradiction that v, for example, is not incident to a red edge. Then the observation before the case distinction implies that w is incident to a red induced P_3 in both F_1 and F_2 which would complete a red induced P_5 which is not possible. Hence both v and w are always incident to a red edge in any good coloring of G. Take two copies of G and connect their v vertices. This graph will function as a blue-determiner. The signal edge is the edge connecting the two copies of G. (There is a good coloring of the blue-determiner, because according to (i) there is a good coloring of G in which both v and w are incident to a red edge, but not a red induced P_3.) As a weak switch we take a copy of G in which we connect v and w by a path of length two (via a new vertex). The two edges of the path are the signal edges e and f. We can use this simple weak switch (instead of a real switch) in the completeness construction, since it will force edges between switches to be blue in good colorings (because we are excluding a red P_4).

In the second case we take two copies F_1, F_2 of F' and identify $v := v_1 = w_2$ and $v_2 = w_1$. If there is a good coloring of the resulting graph G, then G is a strong enforcer with signal vertex v. (Note that both v and w have to be incident with the endpoint of a red induced P_3.) If there is no good coloring, both v and w are always incident to a red edge in every good coloring. Take two copies F_1, F_2 of F'. Try all four combinations of identifying one vertex from v_1, w_1 with one vertex from v_2, w_2. If any of the resulting graphs has a good coloring it is a strong enforcer with any of the two non-identified vertices from $\{v_1, w_1, v_2, w_2\}$ as a signal vertex. In case none of these graphs has a good coloring, both v and w are always incident to the endpoint of a red induced P_3 in any good coloring, so we can simply take F' with v as a signal vertex to be the strong enforcer.

Note that we can effectively decide which case we are in. Case (i) applies if and only if $G \not\rightarrowtail (P_4, K_n)$, and the subcases of case (ii) similarly depend on whether a graph strongly arrows (P_4, K_n).

For the remainder of the proof we can apply Lemma 5.2 in the second case. For the first case we have to carefully check the construction to see that the blue determiners can be used instead of the enforcers, and that they do not interfere with each other.

What have we shown? For the construction of the enforcer we had to make several queries of the form $H \rightarrowtail (P_4, K_n)$. Note, however, that we made only polynomially many queries, and that the queries can be determined before making any of them (since we remove the edges in a fixed order).

This yields a truth-table reduction as follows: fix a sequence of graphs $\emptyset = F_0 \subset F_1 \subset \cdots \subset F_k = F$ such that each graph in the sequence has exactly one more edge than the previous one. Above we showed how to construct for a formula ψ and from a (P_4, K_n)-Ramsey-minimal graph H a graph H' such that ψ is true if and only if $H' \rightarrowtail (P_4, K_n)$. Then ψ is true if and only if for some $0 \leq i < k$ it is true that $F_i \rightarrowtail (P_4, K_n)$, and $F_{i+1} \not\rightarrowtail (P_4, K_n)$, and $F_i' \rightarrowtail (P_4, K_n)$. This condition is a Boolean formula, showing that we have a truth-table reduction. Hence we have established the following result.

Theorem 5.5 *Deciding $F \rightarrowtail (P_4, K_n)$ is truth-table complete for $\mathbf{\Pi_2^p}$.*

This example is meant to illustrate the use of ideas related to Ramsey-minimal graphs. These ideas might also be fruitful in the non-induced case.

6 Other Graphs

In the previous sections we have presented a technique to show that $F \rightarrow (G, K_n)$ and $F \rightarrowtail (G, K_n)$ are $\mathbf{\Pi_2^p}$-complete. This technique works as long as G belongs to a particular class of graphs, namely trees of size at least two.

Since complete graphs are not trees (for $n \geq 3$) this leaves us in the slightly paradoxical situation of being able to show $F \rightarrow (K_{1,2}, K_n)$ complete for $\mathbf{\Pi_2^p}$, but not the seemingly much harder problem $F \rightarrow (K_3, K_n)$. One reason, of course, is that we do not know the Ramsey numbers $r(K_3, K_n)$. We address this problem in the first part of this section by presenting a general result for 3-connected graphs (graphs from which any two vertices can be removed without disconnecting it).

Another question to be asked is whether the complete graph can be substituted by other families of graphs. As we transformed CLIQUE into ARROWING it should be possible to lift similar **NP**-completeness results to $\mathbf{\Pi_2^p}$. A good candidate would be the complete bipartite graphs $K_{n,n}$ [GJ79, Problem GT24]. The use of complete bipartite graphs would significantly simplify the constructions: we could remove the parts that are only concerned with forcing blue edges between x-gadgets. The challenge is the construction of enforcers. In the induced case the results from [SS00] might prove helpful in this regard.

In the second part of this section we sketch a proof that $F \rightarrowtail (P_3, P_n)$ is $\mathbf{\Pi_2^p}$-complete, where P_n is a path on n vertices. This result could be viewed as a lifting of Yannakakis's (unpublished) result that finding induced paths is **NP**-complete [GJ79, Problem GT23]. In our terminology he showed that $F \rightarrowtail (P_2, P_n)$ is **NP**-complete.

6.1 Complete Graphs against Triangles and 3-Connected Graphs

In this section we establish conditions under which $F \rightarrow (G, K_n)$ is $\mathbf{\Pi_2^p}$-complete, if G is a K_3 or a 3-connected graph. For these graphs enforcers cannot exist. Determiners seem to be the natural substitute.

The literature contains constructions of determiners, and negative senders (our switches). See for example Burr's paper [Bur90] which is based on work by Burr, Erdős, and Lovász [BEL76], Burr, Nešetřil, and Rödl [BNR85], and Burr, Faudree, and Schelp [BFS77]. These constructions, however, use exponentially sized hypergraphs, making them unusable for polynomial time.

Perhaps, however, there is a way to build smaller determiners and switches for particular graphs. K_3 would be an obvious first candidate to investigate. Hence we propose the following conjecture:

Conjecture 6.1 *Deciding whether $F \rightarrow (K_3, K_n)$ is $\mathbf{\Pi_2^p}$-complete.*

We will support this conjecture by proving a theorem that reduces the whole problem to the effective construction of determiners.

Let us review the completeness proof of Theorem 4.12. Suppose we have blue determiners (that means the signal edge is forced to be blue). There were two places where we needed blue enforcers: to build switches and to force edges to be blue. The second task can be accomplished by blue determiners now. We just take one such determiner for each edge we want to be forced to be blue. Since we restrict ourselves to 3-connected graphs the determiners work as expected as the graphs cannot extend over a single edge.

We are left with the construction of switches. Lemma 2.3 in Burr, Nešetřil, and Rödl [BNR85] shows how to build a switch in polynomial time given a (G, H)-determiner and a (H, G)-determiner for some 3-connected G and H. Hence all we are left with is the construction of a (H, G)-determiner from a (G, H)-determiner. Take a copy of G and make each edge except for one (call it e) the signal edge of a seperate (G, H)-determiner. In any (H, G)-good coloring all the signal edges have to be blue and hence e is forced to be red. Since G and H are 3-connected it is clear that a monochromatic copy of G or H cannot extend beyond the identified edges. This means that a monochromatic copy of G or H is either contained in one of the determiners or in the copy of G. (The same is true for a K_3.) Hence a (H, G)-good coloring will result from merging the (H, G)-good colorings of the determiners and making e red.

Note that these switches are much simpler than the ones we used in the first sections. This is because there we had to spend some ingenuity on forcing the edges between x-gadgets to be blue. Now that we are dealing with 3-connected graphs, we can simply use blue determiners to achieve that task.

The above discussion can be summed up in a conditional result.

Theorem 6.2 *Let G be a fixed 3-connected graph or a K_3. If a (strong) (G, K_n)-determiner can be constructed in polynomial time, then deciding $F \rightarrow (G, K_n)$ ($F \rightarrowtail (G, K_n)$) is $\mathbf{\Pi_2^p}$-complete.*

Building a determiner for (K_3, K_n) is probably a hard task. As a first approach one could try to apply the ideas related to Ramsey-minimal graphs presented in the preceding section.

6.2 Induced Paths

So far all the completeness results were for pairs of graphs including the complete graph. The reason is that this allows us to look at arrowing as coding any uniform family of clique problems. Since the clique problem is **NP**-complete this gives us the completeness result on the second level. In this section we show that we can similarly lift the **NP**-complete problem of testing for an induced P_n to the second level.

We believe that the problem $F \rightarrowtail (P_k, P_n)$ is $\mathbf{\Pi_2^p}$-complete for any fixed k. We settle for the modest goal of proving this statement for $k = 3$.

Theorem 6.3 *Deciding $F \rightarrowtail (P_3, P_n)$ is $\mathbf{\Pi_2^p}$-complete.*

We will show the theorem in two steps. We first prove the existence of a strong enforcer, and then show how to use the gadget in the completeness construction.

Lemma 6.4 *We can build a strong (P_3, P_n)-enforcer in polynomial time in n.*

Proof. We will show how to build a strong (P_3, P_{2n})-enforcer, the other case is similar. Take a $K_{2,3}$ and let a and b be its two vertices of degree 3. Then in any coloring of the $K_{2,3}$ that does not contain a red P_3, there is a blue (induced) path of length 2 from a to b. Take $n-1$ such $K_{2,3}$'s and line them up by identifying the b vertex of one $K_{2,3}$ with the a vertex of the next. Add a vertex v and connect it to the a vertex of the first $K_{2,3}$. Call the resulting graph G. It is easy to see that if G does not contain a red induced P_3 it will contain a blue induced P_{2n-1} starting with the a vertex of the first $K_{2,3}$. Hence the edge incident with v is forced to be red, which makes v a signal vertex of G. A (P_3, P_{2n})-good coloring is obtained by coloring the edge incident with v red, and the remainder of G blue. $\diamondsuit$

Lemma 6.5 *For a given* Π_2 *sentence* ψ *we can construct a graph* F *and compute a number* n *in polynomial time such that* ψ *is true if and only if* $F \rightarrowtail (P_3, P_n)$.

Proof. Let $\psi = (\forall x_1, \ldots, x_l)(\exists y_1, \ldots, y_k)[\varphi(x_1, \ldots, x_l, y_1, \ldots, y_k)]$ be the formula to be coded with its $l + k$ variables ranging over $\{0, 1\}$. We will refer to the x_i as the *x-variables* and the y_i as the *y-variables*. We make the usual assumptions on φ, that is that it is in conjunctive normal form with at most three literals per clause, and each literal occurs at most twice in the whole formula. As before x-variables occur in exactly two clauses: $(x \vee \overline{y})$ and $(\overline{x} \vee y)$. Let the total number of clauses be c.

The basic coding device for y-variables is built from a $K_{3,3}$ by identifying its vertices in order with every other vertex of a C_{12}. Call this graph S_y. We call the two vertices of S_y which are in a triangle and have degree two I_y and O_y. Number the vertices of S_y which belong to C_{12} in order such that I_y is number one and O_y is number 7 (see Figure 8).

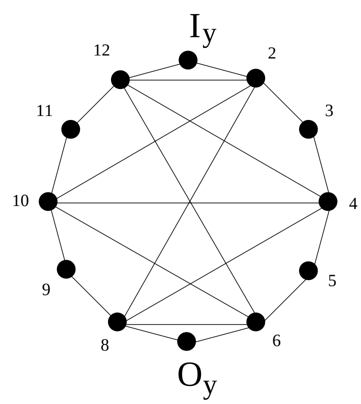

Figure 8: A y-gadget

For every clause C we take a new edge e_C, and for every x-variable, two vertices u_x and v_x. From these build a graph F' as follows. We start with u_{x_1} and connect it to one vertex of e_C by a P_3 where C is the clause that x_1 appears in. Connect the other vertex of e_C to v_{x_1} by another P_3. Do the same for the clause C' in which $\overline{x}_1$ appears. Now connect v_{x_1} by an edge to u_{x_2} and repeat the procedure until we have reached v_{x_l} which we will then connect by an edge to I_{y_1}. Identify O_{y_i} with $I_{y_{i+1}}$ for all $i < k$. For each y_i that occurs in a clause C we take two vertices from S_{y_i} (either number 2 and 4, or number 4 and 6) and connect one of the vertices to one end of e_C by a P_3 and the other vertex to the other end by another P_3. For $\overline{y}$ we can take either number 8 and 10 or number 10 and 12. Since each literal occurs at most twice this is sufficient.

Finally, from F' we get the graph F by attaching blue $(P_3, P_{3l+6k+c})$-enforcers to every vertex in an S_y, every v_x, and every vertex belonging to some e_C. (These are all the vertices in F except for the u_x vertices, and the middle vertices of the paths of lengths two leading to some e_C.)

For an example of what F' looks like (that is, F without the enforcers) see Figure 9.

We claim that $F \rightarrowtail (P_3, P_{3l+6k+3c})$ if and only if ψ is true.

Suppose that ψ is true, but there is a coloring of F which contains no induced red P_3 and no induced blue $P_{3l+6k+3c}$. Fix such a coloring. In particular all the enforcers work as they should and force all the

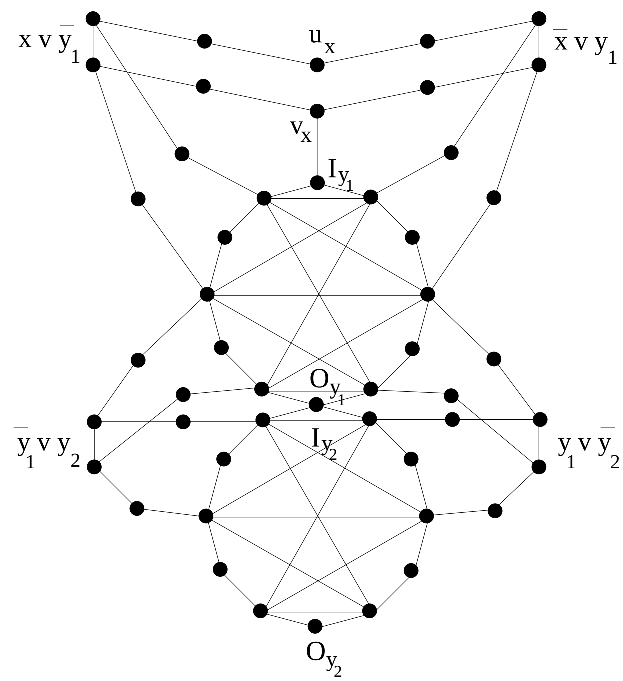

Figure 9: F' for $(\forall x)(\exists y)[x \not\leftrightarrow y] \equiv (\forall x)(\exists y_1, y_2)[(x \vee \overline{y}_1) \wedge (\overline{x} \vee y_1) \wedge (y_1 \vee y_2) \wedge (\overline{y}_1 \vee \overline{y}_2)]$

edges in F' that are incident with an enforcer to be blue. In particular, for all x-variables there will be a blue path of length five from u_x to v_x (since u_x can have at most one outgoing red edge), and hence there will be a blue path of length six from each u_{x_i} to $u_{x_{i+1}}$ (or I_{y_1}). Depending on whether the path runs through the x clause or the $\overline{x}$ clause, call x true or false, respectively. With this assignment to the x-variables there is an assignment to the y-variables such that $\varphi(x_1, \dots, x_l, y_1, \dots, y_k)$ is true. Fix such an assignment. For each clause C choose a y-variable that makes it true, unless the clause is already made true by the assignment to the x-variables. There are $c - l$ such clauses. We can now extend the blue path of length $6l$ from u_{x_1} to I_{y_1} through the y-variable coding devices and the $c - l$ clauses that remain to be satisfied. This will add $6k + 3(c - l)$ edges to the path (since every detour through a clause that has to be satisfied adds three edges to the total length). This gives us an induced blue path of length $3l + 6k + 3c$ contradicting the assumption.

Suppose now that $F \rightarrowtail (P_3, P_{3l+6k+3c})$, but ψ is false, that is, there is a truth-assignment to the x-variables (which we fix) such that $\varphi(x_1, \dots, x_l, y_1, \dots, y_k)$ is false for all assignments to the y-variables. Pick a coloring of F as follows. Choose a good coloring for each enforcer and color all edges of F' which are incident to the signal vertices blue. For each x-variable, color blue the edge from u_x which is on the path to the clause containing the x-literal that is true, and red the edge from u_x leading to the other clause containing an x-literal.

By assumption there is an induced blue path of length $3l + 6k + 3c$. The path cannot go through any of the enforcers, since they are linked to F' by a red edge (and the enforcers themselves do not contain paths of length $3l + 6k + 3c$). Each S_y can contain at most six edges of an induced blue path, and each x coding device (with the corresponding clauses) at most six edges. Hence there are at least $(3l + 6k + 3c) - (6k + 6l) = 3(c - l)$ blue edges of the path left which therefore have to correspond to the edges in the remaining $c - l$ clauses that are not already satisfied by the assignment to the x-variables. For each of these $c - l$ clauses pick a y-variable which is directly connected to it on the blue path, and call that y-variable true or false according to whether it is connected to the positive or negative side of the y-coding device. The way F' was constructed assures that this gives us a consistent assignment of truth-values to some y-variables. (The assignment is consistent because the $K_{3,3}$ makes it impossible for an induced path to run through both the positive and negative side of a y-coding device.) Furthermore, this assignment makes all clauses true for the particular values of x chosen, which contradicts the assumption. $\diamondsuit$

7 The Diagonal Case

If we restrict the questions $F \rightarrow (G, H)$ and $F \rightarrowtail (G, H)$ to identical G and H we get a problem of independent interest, called the diagonal case. In Ramsey theory one usually writes $F \rightarrow G$ for $F \rightarrow (G, G)$ and $F \rightarrowtail G$ for $F \rightarrowtail (G, G)$. We will, however, adhere to our binary notation.

What then is the complexity of the diagonal case, namely deciding $F \rightarrow (G, G)$ and $F \rightarrowtail (G, G)$? We are confronted with two phenomena that we did not encounter in the more general problem. Consider, for example, the complexity of $F \rightarrow (K_n, K_n)$. If $n \geq 2 \log |F|$, then $F \rightarrow (K_n, K_n)$ is false, since F is too small to contain a monochromatic K_n in every coloring (remember that $r(n) \geq 2^{n/2}$). Hence the complexity of $F \rightarrow (K_n, K_n)$ sits in $\mathbf{coNP}^{\mathsf{LOGCLIQUE}}$ where $\mathsf{LOGCLIQUE}$ is the problem of deciding whether a graph F has a clique of size at least $\log |F|$. We expect $\mathbf{coNP}^{\mathsf{LOGCLIQUE}}$ to be smaller than $\mathbf{\Pi_2^p}$. The second observation is that we cannot make use of determiners anymore, as they require asymmetry.

There is not much we can say about the complexity of the non-induced diagonal case at this point. It is at least as hard as $\mathbf{coNP}$ (remember that $F \rightarrow (K_3, K_3)$ is $\mathbf{coNP}$-complete). Since we showed that $F \rightarrow (P_3, P_n)$ is $\mathbf{\Pi_2^p}$-complete, paths might be a good candidate for a $\mathbf{\Pi_2^p}$-completeness proof.

Conjecture 7.1 *$F \rightarrow (P_n, P_n)$ is $\mathbf{\Pi_2^p}$-complete.*

Let us have a closer look at $F \rightarrow (K_n, K_n)$ before we study the induced case. As mentioned above the problem lies in $\mathbf{coNP}^{\mathsf{LOGCLIQUE}}$. $F \rightarrow (K_n, K_n)$ if and only if for all subgraphs F' of F either F' or $F - F'$ contains a clique of size n. This is equivalent to saying that for all subgraphs F' of F the disjoint union of F' and $F - F'$ contains a clique of size n. Hence we need only one query, and the problem lies in $\mathbf{coNP}^{\mathsf{LOGCLIQUE}[1]}$, that is, one many-one access to the $\mathsf{LOGCLIQUE}$ problem along each computation path of a $\mathbf{coNP}$ machine (checking all subgraphs of F). Assuming that switches (negative senders) for K_n can be constructed in exponential time (the construction of negative senders in [BNR85] does not even seem to be primitive recursive), the problem can be shown to be complete for problems of the following form: for all x either $f(x)$ or $F - f(x)$ contains a clique of size at least n, where f computes a graph in polynomial time from x, and F is a graph. As we saw above this class of problems is a subclass of $\mathbf{coNP}^{\mathsf{LOGCLIQUE}[1]}$, and likely a proper subclass. It is not clear whether it has a natural characterization in terms of complexity classes.

How about the induced case? Interestingly, here stars will work. Remember that the non-induced version $F \rightarrow (K_{1,n}, K_{1,m})$ can be decided in polynomial time [Bur90]. Yannakakis and Lewis [GJ79, Problem GT21] showed that deciding whether a graph has a $K_{1,n}$ induced subgraph is $\mathbf{NP}$-complete (they actually proved a more general result). This implies that $F \rightarrowtail (K_{1,1}, K_{1,n})$ is $\mathbf{NP}$-complete. It is easy to see this directly: a graph G has an independent set of size at least n if and only if the graph obtained from G by adding a vertex and connecting it to all vertices of G contains an induced $K_{1,n}$ (for $n > 4$, remember that we can assume that G has maximal degree 3). Hence $F \rightarrowtail (K_{1,n}, K_{1,1})$ is $\mathbf{NP}$-complete.

Theorem 7.2 *$F \rightarrowtail (K_{1,n}, K_{1,n})$ is $\mathbf{\Pi_2^p}$-complete.*

Proof. Let $\psi = (\forall x_1, \ldots, x_k)(\exists y_1, \ldots, y_k)[\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)]$. We are making the same assumptions as in the earlier proofs. In particular, each x variable will appear in exactly two clauses, namely $(x \vee \overline{y})$ and $(\overline{x} \vee y)$.

Let $C_1, \ldots, C_n$ be the clauses of φ. Construct a graph G as follows. For each clause C_i take as many vertices as there are literals in the clause, connect them, and label them with the corresponding literals. Connect two vertices belonging to different clauses if and only if their labels are contradictory. Then G will contain an independent set of size n if and only if $(\exists x_1, \ldots, x_k)(\exists y_1, \ldots, y_k)[\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)]$ is true. Note that G will have maximum degree at most 4.

Let G' be the graph constructed from G by taking a new vertex w and connecting it to all vertices in G. Then G' will contain an induced $K_{1,n}$ if and only if $(\exists x_1, \ldots, x_k)(\exists y_1, \ldots, y_k)[\varphi(x_1, \ldots, x_k, y_1, \ldots, y_k)]$ is true (assuming $n > 4$).

Construct a graph B_n as follows. Start with a copy of $K_{2,2n-3}$ and call the two vertices in the smaller partition u and v (we assume that $n \geq 3$). Insert an edge between u and v, and for each vertex of the $2n-3$ nodes of the larger partition of the $K_{2,2n-3}$, take a copy of a $K_{1,2n-3}$ and identify the node with the center of that star. See Figure 10.

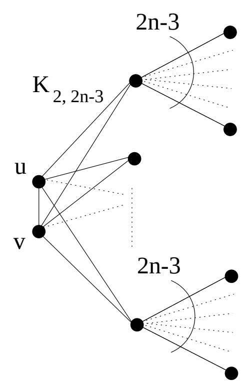

Figure 10: A positive sender for stars

We claim that B_n acts as a positive sender with signal vertices u and v in the following sense: in any $(K_{1,n}, K_{1,n})$-good coloring of B_n additional edges out of u and v are all forced to have the same color, and there is a good coloring in which edges out of u and v are forced to be red, and one in which they are forced to be blue.

For the first half of the claim consider a good coloring of B_n. Each of the $2n-3$ stars $K_{1,2n-3}$ we added to the bipartite graph will have $n-1$ red edges and $n-2$ blue edges, or vice versa. This forces the pair of edges out of each center to u and v to have the same color. Hence there must be $n-1$ red pairs and $n-2$ blue pairs, or vice versa. Suppose the first. Then u and v are each at the center of a red induced $K_{1,n-1}$, and hence all additional edges out of u and v (even if adjacent) are forced to be blue. We can construct a good coloring as follows. Suppose we want to force all edges out of u and v to be blue. Look at the $2n-3$ vertices in the second partition of the $K_{2,2n-3}$, and the pairs of edges leaving them towards u and v. Color $n-1$ of these pairs of edges red, and the other $n-2$ pairs blue. Fix corresponding colorings of the $2n-3$ stars centered in the vertices of the second partition: if the center of a $K_{1,2n-3}$ is incident with a red pair, make $n-2$ of the edges of the star red, and the remaining $n-1$ blue, and vice versa. Note that the edge between u and v assures that this is a good coloring since an induced $K_{1,n}$ cannot include both u and v as outer vertices.

From B_n we construct a family of graphs $D_{n,m}$ as follows. Start with a K_m and m copies of B_n. Identify each vertex of K_m with a signal vertex of a B_n. By construction the m non-identified signal vertices will force all additional edges leaving these m signal vertices to have the same color. We can use $D_{n,m}$ to construct a switch-like graph which we will call a reverser for the purposes of this proof. Reversers relate to switches as enforcers do to determiners. Start with a copy of a $D_{n,n(n-1)}$, and n copies of $K_{1,n-1}$. Identify the non-identified signal vertices of $D_{n,n(n-1)}$ with the $n(n-1)$ outer vertices of the n stars. Add another $K_{1,n-1}$ and identify its outer vertices with $n-1$ of the n center vertices from the stars added earlier. Call the two non-identified center vertices u and v. This graph acts as a reverser with signal vertices u and v in the following sense. Edges out of u will be forced to have the opposite color of edges out of v, and there are good colorings in which edges out of u are red, and edges out of v blue, and vice versa.

The idea used in the construction of the reverser we can also use to force edges in G to have the same color. Construct a graph D_n as follows. Take a copy of $D_{n,3n(2n-3)}$. Add $3n$ copies of $K_{1,n-1}$ and $3n$ copies of $K_{1,n-2}$. Identify the $3n(2n-3)$ signal vertices of $D_{n,3n(2n-3)}$ with the outer vertices of the stars. The center vertices of the stars fall into two categories depending on whether they are at the center of a $K_{1,n-1}$, or a $K_{1,n-2}$. Call the former y-signal vertices, and the latter x-signal vertices of D_n.

We are now ready to construct F. Take G' and a copy of D_n. Consider the vertices of G (as a subgraph of G'): make those labeled with a y-literal y-signal vertices of D_n, and those labeled with an x-literal x-signal vertices of D_n. Since there are at least $3n$ signal vertices of each type, there will be sufficiently many signal vertices to accomplish this. Finally, for each x-variable add a reverser, and two edges connecting the two vertices in G corresponding to the x-variable to the two signal vertices of the reverser. This completes the construction of F.

Note that in a good coloring all edges in G are forced to have the same color, say blue. This is immediate for edges incident with a y-signal vertex. The only other case is an edge uv labeled with contradictory x-literals. If uv were red, it would complete a red induced $K_{1,n-1}$ with center in u. Hence all other edges leaving u are forced to be blue, in particular the one to the reverser, which forces the edge connecting the reverser to v to be red, completing a red induced $K_{1,n-1}$ in v, forcing all other edges out of v, and in particular uv, to be blue, which is a contradiction. Hence all edges in G are forced to have the same color.

Furthermore, a very similar argument shows that if u and v are two vertices in G labeled with contradictory x-literals, then at least one of uw or vw is blue (where w is the vertex of G' which is adjacent to every vertex in G). A red uw, say, would complete a red induced $K_{1,n-1}$ with center in u forcing all other edges leaving u to be blue, including the one to the reverser. This in turn forces the edge connecting the reverser to v to be red, completing a red induced $K_{1,n-1}$ in v, forcing all other edges out of v, and in particular vw, to be blue.

We claim that $F \rightarrowtail (K_{1,n}, K_{1,n})$ if and only if ψ is true.

Suppose that ψ is true, but $F \not\rightarrowtail (K_{1,n}, K_{1,n})$. Fix a $(K_{1,n}, K_{1,n})$ good coloring of F. Since the coloring is good, all the gadgets work as expected. This means all edges in G have the same color (without loss of generality this color is blue), edges out of w to vertices in G which do not correspond to x-literals are also blue, and out of the two edges from w to contradictory x-literals at least one is blue. For each x variable x_i say it is true if the edge from w to the vertex labeled x_i is blue, and false otherwise (in which case there is a blue edge from w to the vertex labeled with the negation of x_i). For this particular assignment choose an assignment of truth-values to the y variables such that

$\varphi(x_1,\dots,x_k,y_1,\dots,y_k)$ is true. By the construction of G there are n vertices in G which form an independent set, and all of them are connected to w by blue edges, hence F contains a blue induced $K_{1,n}$ contradicting the assumption.

To show the other direction assume that $F \rightarrowtail (K_{1,n}, K_{1,n})$ and ψ is false. Fix a truth-assignment to the x variables such that $\varphi(x_1,\dots,x_k,y_1,\dots,y_k)$ is false for all truth-assignments to the y-variables. We have to construct a good coloring for F. Make all edges in G blue, and extend this with a good coloring of D_n in which its underlying complete graph is red. Then the x-signal vertices are at the center of a red induced $K_{1,n-2}$, and the y-signal vertices are at the center of a red induced $K_{1,n-1}$. Color an edge from w to a y-vertex blue, and edges from w to a vertex labeled with an x literal blue if that literal is true, and red otherwise. Extend this with a good coloring of the reversers and the edges connecting them to G.

We claim that an induced monochromatic $K_{1,n}$ can only occur with w as the center. Since we chose good colorings of all the gadgets (D_n, and the reversers), none of them will contain an induced monochromatic $K_{1,n}$. The two signal vertices of a reverser are incident with a red $K_{1,n-1}$ and one blue edge, or a blue $K_{1,n-1}$, and a red edge, hence they are not at the center of an induced monochromatic $K_{1,n}$. By choice of the coloring both the x-vertices and the y-vertices in G are incident with $n-1$ red edges, and at most five blue edges (since G has maximum degree four, at most four edges from G, and one to w). Hence if $n > 5$ they cannot be at the center of a red induced $K_{1,n}$. This leaves only w.

If there is an induced monochromatic $K_{1,n}$ centered in w it cannot be red, since there is at most one red edge from w to each clause, and there is a clause to which there is no red edge from w. Hence there must be a blue induced $K_{1,n}$ with center in w with its outer vertices in G. Since the $K_{1,n}$ is induced its outer vertices form an independent set in G of size n. By the construction of G this set corresponds to an assignment to the x and y variables that makes φ true. Moreover the way we chose the coloring ensured that this assignment is consistent with the given assignment to the x-variables. Therefore $\varphi(x_1,\dots,x_k,y_1,\dots,y_k)$ is true contradicting the choice of the assignment of truth values to the x-variables. $\diamondsuit$

8 Other Complexities

In the preceding section we discussed the complexity of $F \rightarrow (K_n, K_n)$ and why we do not expect it to be $\mathbf{\Pi_2^p}$-complete. The simplest examples of arrowing problems that are not $\mathbf{\Pi_2^p}$-complete (unless **PH** collapses) occur if G (or H) is a single edge. $F \rightarrow (P_2, H)$ is equivalent to H being a subgraph of F (if F contains at least one edge). Similarly $F \rightarrowtail (P_2, H)$ is equivalent to H being an induced subgraph of F (if F contains at least one edge). Hence these problems lie in **NP**, and several are **NP**-complete, since the subgraph problems are. $F \rightarrow (P_2, K_n)$, $F \rightarrow (P_2, K_{n,n})$, $F \rightarrow (P_2, P_n)$, and $F \rightarrow (P_2, C_n)$ correspond to the clique problem, complete balanced bipartite subgraph, the longest path problem, and Hamiltonian cycle, and are therefore **NP**-complete. Similarly in the case of induced arrowing $F \rightarrowtail (P_2, P_n)$, and $F \rightarrowtail (P_2, K_{1,n})$ are **NP**-complete. The first problem corresponds to finding induced subpaths [GJ79, Problem GT23], and the second problem is **NP**-complete by the first paragraphs in the proof of Theorem 7.2.

If both G and H are fixed graphs, then the problem lies in **NP**, and can indeed be **NP**-complete, as witnessed by $F \rightarrow (K_3, K_3)$ [GJ79, Problem GT6]. There are cases, however, in which the problem lies in **P**, as observed by Burr [Bur90]. If the set of Ramsey-minimal graphs of G and H is finite, then $F \rightarrow (G, H)$ can be solved in polynomial time, since we only have to check F for a finite number of subgraphs. As an example Burr mentions $F \rightarrow (nK_2, H)$, where both n and H are fixed. It is not clear, however, whether we can determine all the Ramsey-minimal graphs effectively, even if we know that there are only finitely many. (On a smaller scale this seems to mirror the situation in the graph minor theorem.)

For our final example we look at stars: $F \rightarrow (K_{1,n}, K_{1,m})$. Burr showed that the problem $F \not\rightarrow (K_{1,n}, K_{1,m})$ m-reduces to PERFECT MATCHING [Bur90] in logarithmic space. A *perfect matching* of a graph G is a subset of the edges of G such that every vertex of G is incident to exactly one edge of the subset. PERFECT MATCHING is the corresponding decision problem asking whether a graph G has a perfect matching. We will show that PERFECT MATCHING log-space m-reduces to $F \not\rightarrow (K_{1,n}, K_{1,m})$ which means that these two problems have the same complexity (up to log-space m-reductions). This implies that determining the precise complexity of the arrowing problem is going to be difficult, since little is known about the matching problem. We do know that it lies in both **P** and **RNC** (randomized circuits of polylogarithmic depth and polynomial size). Since **RNC** and **P** are expected to be incomparable (see [GHR95]), it is unlikely that the arrowing problem will turn out to be complete for either.

Theorem 8.1 *$F \not\rightarrow (K_{1,n}, K_{1,m})$ and* PERFECT MATCHING *are logarithmic-space m-equivalent.*

Proof. Burr [Bur90] showed that the arrowing problem reduces to a matching problem, which in turn reduces to PERFECT MATCHING. Hence it will be sufficient to show that PERFECT MATCHING m-reduces to $F \not\rightarrow (K_{1,2}, K_{1,m})$.

Given an integer $d > 2$ construct a graph E'_d as follows. Take a copy of $K_{d-1,d-1}$ and three vertices u, v, and w. Connect u by single edges to the $d-1$ vertices in first partition of $K_{d-1,d-1}$, and w to the $d-1$ vertices in the second partition. Finally, add edges uv and wv to get E'_d. We claim that in a $(K_{1,2}, K_{1,d})$-good coloring of E'_d both uv and wv are forced to be blue. For a contradiction suppose there was a good coloring in which uv, say, was red. Then the $d-1$ edges out of v to the first partition are forced to be blue. Since every vertex in the first partition is connected to the $d-1$ vertices in the second partition, at least one of these edges has to be red (to avoid a blue $K_{1,d}$ with the edge to u). Furthermore, at most one edge is red, since we have to avoid red $K_{1,2}$s. Hence the red edges between the first and second partition form a matching of the vertices in the two partition. Hence all the edges from the second partition to w are forced to be blue, forcing wv to be red. This, however, completes a red $K_{1,2}$ with center in v.

On the other hand E'_d does have a good coloring: make uv and wv blue, make one of the edges from u to the first partition red, the others blue, do the same thing for v and the second partition, connect the red neighbors of u and v by a blue edge, fix a perfect red matching for the $2d-2$ blue neighbors of u and v, and color the remaining edges blue.

Assume that d is odd. We construct E_d from E'_d. Take $(d-1)/2$ copies of E'_d and identify their v-vertices. Add two new vertices s and t, and edges st and tv. In any good coloring tv will be forced to be red (since v is incident with $2(d-1)/2 = d-1$ blue edges), and hence st is forced to be blue.

With E_d we can now describe the reduction. Let G be the graph for which we want to decide whether it has a perfect matching. Let d be the smallest odd integer larger than the maximum degree of G. For each vertex v of G take $d - degree(v)$ copies of E_d, and identify v with the s-vertices of the E_d. Call the resulting graph F. We claim that G has a perfect matching if and only if $F \not\rightarrow (K_{1,2}, K_{1,d})$. If there is a good coloring of F, then all the gadgets work, and every vertex of G in F is incident with d edges, exactly one of which must be red. This red edge must lie in G, since the additional edges we added to vertices in G are all forced blue. Hence the red edges form a perfect matching of G. If we are given a perfect matching of G, we color the edges of G which belong to the matching red (in F), the edges in G not in the matching blue (in F), and fix good colorings for all gadgets. $\diamondsuit$

9 Conclusion

We have presented several completeness results for both the induced and the non-induced case, and derived some general results that reduce the problem to the effective construction of graphs like (strong) enforcers and (strong) determiners. These constructions are combinatorial problems of some trickiness. In a first step the paper [SS00] collects effective constructions of graphs that strongly arrow other graphs, but more work is to be done.

The most interesting task left by this paper is the construction of a (K_3, K_n)-determiner (with or without oracle help). By Theorem 6.2 this would imply the $\mathbf{\Pi_2^p}$-completeness of $F \to (K_3, K_n)$. Another question we would like to settle is the complexity of diagonal arrowing. It seems that $F \to (P_n, P_n)$ should be a good candidate for further investigation.

The two tables collect the results known (to me) on arrowing and strong arrowing, excluding the conditional results proved in this paper, and the trivial **NP**-completeness results (where $G = P_2$).

Relation	Fixed Parameters	Computational Complexity	Reference
$F \to (T, K_n)$	tree T, $\lvert T\rvert \geq 3$	complete for $\mathbf{\Pi_2^p}$	Theorem 4.1
$K_n \to (G, H)$		hard for **NP**	Burr [Bur84a]
$F \to (K_3, K_3)$		complete for **NP**	Burr [GJ79]
$F \to (nK_2, H)$	n, H	in **P**	Burr [Bur90]
$F \not\to (K_{1,n}, K_{1,m})$		PERFECT MATCHING	Burr [Bur90], Theorem 8.1

Table 1: Complexity of Arrowing

Relation	Fixed Parameters	Computational Complexity	Reference
$F \rightarrowtail (K_{1,p}, K_n)$	$p \geq 2$	complete for $\mathbf{\Pi_2^p}$	Theorem 5.1
$F \rightarrowtail (K_{1,n}, K_{1,n})$		complete for $\mathbf{\Pi_2^p}$	Theorem 7.2
$F \rightarrowtail (P_3, P_n)$		complete for $\mathbf{\Pi_2^p}$	Theorem 6.3
$F \rightarrowtail (P_4, K_n)$		tt-complete for $\mathbf{\Pi_2^p}$	Theorem 4.1

Table 2: Complexity of Strong Arrowing

A combinatorial problem related to $F \to (K_{1,n}, K_{1,m})$ is finding a characterization of graphs F for which $F \to (K_{1,n}, K_{1,m})$. Such a characterization is known for the diagonal case ($n = m$), but even the case $F \to (K_{1,2}, K_{1,3})$ is open and seems difficult [BEL76].

Another combinatorial challenge is hidden in the completeness result of the induced diagonal case. The result implies that there are polynomial-time computable functions f and g such that $F \rightarrowtail (G, H)$ if and only if $f(F, G, H) \rightarrowtail (g(F, G, H), g(F, G, H))$, that is, the general case can effectively be reduced to the diagonal case. Can f and g be substituted by (effective) combinatorial constructions?

The result for stars suggests a generalization. Is it true that $F \to (G, H)$ lies in **P**, if G and H are trees of bounded diameter? Obviously we cannot relax the condition that G and H be trees, since $F \to (K_3, K_3)$ is already **NP**-complete.

The effectiveness of general Ramsey theory (as opposed to graph Ramsey theory) is covered in a survey by Bill Gasarch [Gas98]. Most of the results in this area belong to computability rather than complexity. There is, however, one exception which moreover establishes a link to graph Ramsey Theory, and that is the computation of Ramsey numbers. Remember the definition of the generalized Ramsey number of two graphs G and H as $r(G,H) = \min\{n : K_n \to (G,H)\}$. The usual Ramsey numbers can be defined from this as $r(k,l) = r(K_k, K_l)$. The question of how hard it is to compute $r(k,l)$ is probably not a good one, since $r(k,l)$ might be exponentially large compared to k and l, so that an algorithm in the polynomial hierarchy might fail just because of the size of the output, whereas we would not expect the problem to be hard for exponential time. This anomaly disappears if we consider the graph variant: $K_m \to (G,H)$. Burr [Bur84a] considered this problem and showed that it is **NP**-hard. That means that deciding $r(G,H) < m$ is **NP**-hard (where m, in unary, is part of the input). Since the best upper bound we have is $\mathbf{\Pi_2^p}$, this leaves us with a gap which will be more difficult to close than the one for the ARROWING problem, since we rely heavily on the structure of the graph F we are constructing. The situation for $r(k,l)$ seems worse. Burr's proof requires one of G or H to be a path, so the restriction to complete graphs necessitates new ideas. There do not seem to be any lower bounds on the complexity of computing $r(k,l)$.

We should also mention an analogue of $F \to (G,H)$ in computability. If the edge set of (the possibly infinite graph) F is computably enumerable, and G and H are finite, then ARROWING m-reduces to $\overline{\emptyset'}$, the complement of the halting problem (if $e_1, \dots, e_n, \dots$ is an enumeration of the edges of F ask whether there is a (G,H)-good coloring of F restricted to edges $e_1, \dots, e_n$ for all n). This observation is complemented by a result of Burr [Bur84b] who showed that $\overline{\emptyset'}$ can be m-reduced to ARROWING even if F is restricted to be a highly computable (essentially finite and periodic) graph. Hence this infinite version of ARROWING is $\mathbf{\Pi_1}$-complete.

In conclusion, let us point out some other results inquiring into the complexity of Ramsey theory. Burr showed that deciding whether a finite set of points in the plane can be colored with three colors such that no two points within distance K have the same color is **NP**-complete [Bur82]. This seems to be the only computational result in Euclidean Ramsey theory which considers pointsets in finite-dimensional real space. One obvious question to ask is: does Burr's problem become $\mathbf{\Sigma_2^p}$-complete if K is not restricted to be fixed?

Acknowledgments My thanks go to Gerry Brady, Janos Simon, Amber Settle, and Pradyut Shah for their patience in listening to earlier and unnecessarily complicated versions of the proofs. I am also grateful to Lance Fortnow for pointing out several flaws in the original argument, and to Steve Fenner for pointing out a gap in the simplified argument. Thanks go to the anonymous referees for several corrections, in particular of my confused notion of induced arrowing. And thanks again to Amber for careful proofreading.

Thanks to Stuart Kurtz for his optimistic encouragement, to Laci Babai for patiently answering my questions, to Tom Hayes for coming up with the n-partite construction, and to Holger Petersen for some historical comments. Dieter van Melkebeek deserves special thanks for several remarks that greatly simplified the proofs of the main theorems. Finally, thanks go to Martin Kummer for getting me interested in Ramsey theory in the first place.

References

[AT96] Manindra Agrawal and Thomas Thierauf. The Boolean Isomorphism Problem. In *37th Annual Symposium on Foundations of Computer Science*, pages 422–430, Burlington, Vermont, 14–16 October 1996. IEEE.

[BEL76] Stefan A. Burr, Paul Erdős, and Laszlo Lovász. On Graphs of Ramsey Type. *Ars Combinatoria*, 1:167–190, 1976.

[BFS77] Stefan A. Burr, R. J. Faudree, and R. H. Schelp. On Ramsey-minimal Graphs. In *Proceedings of the 8th Southeastern Conference on Combinatorics, Graph Theory and Computing*, number XIX in Congressus Numerantium, pages 115–124, Winnipeg, 1977.

[BNR85] Stefan A. Burr, Jaroslav Nešetřil, and Vojtech Rödl. On the Use of Senders in Generalized Ramsey Theory of Graphs. *Discrete Mathematics*, 54:1–13, 1985.

[Bur82] Stefan A. Burr. An NP-complete Problem in Euclidean Ramsey Theory. *Congressus Numerantium*, 35:131–138, 1982.

[Bur84a] Stefan A. Burr. Determining Generalized Ramsey Numbers is NP-hard. *Ars Combinatoria*, 17:21–25, 1984.

[Bur84b] Stefan A. Burr. Some undecidable problems involving the edge-coloring and vertex-coloring of graphs. *Discrete Mathematics*, 50:171–177, 1984.

[Bur87] Stefan A. Burr. What Can We Hope to Accomplish in Generalized Ramsey Theory? *Discrete Mathematics*, 67:215–225, 1987.

[Bur90] Stefan A. Burr. On the Computational Complexity of Ramsey-Type Problems. In Nešetřil & Rödl, editor, *Mathematics of Ramsey Theory.* Springer-Verlag, 1990.

[CH72] Václav Chvátal and Frank Harary. Generalized Ramsey Theory for Graphs, III. Small Off-Diagonal Numbers. *Pacific Journal of mathematics*, 41(2):335–345, 1972.

[Chv77] Václav Chvátal. Tree-Complete Graph Ramsey Numbers. *Journal of Graph Theory*, page 93, 1977.

[Die97] Reinhard Diestel. *Graph Theory.* Springer, New York, 1997.

[Gas98] William Ian Gasarch. A Survey of Recursive Combinatorics. In Ershov, Goncharov, Nerode, Remmel, and Marek, editors, *Handbook of Recursive Mathematics: Recursive Algebra, Analysis and Combintatorics*, volume II. North–Holland Publishing Co., 1998.

[GHR95] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979.

[GRS90] Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory.* Wiley, 1990.

[Hay] Thomas Hayes. Personal communication.

[HNR83] Frank Harary, Jaroslav Nešetřil, and Vojtech Rödl. Generalized Ramsey Theory for Graphs XIV: Induced Ramsey Numbers. In Miroslav Fiedler, editor, *Graphs and Other Combinatorial Topics*, pages 90–100. Teubner, 1983.

[Huy84] D. T. Huynh. Deciding the Inequivalence of Context-free Grammars with 1-Letter Terminal Alphabet is Σ_2^p-complete. *Theoretical Computer Science*, 33(2-3):305–326, October 1984.

[HW97] Edith Hemaspaandra and Gerd Wechsung. The Minimization Problem for Boolean Formulas. In *38th Annual Symposium on Foundations of Computer Science*, pages 575–584, Miami Beach, Florida, 20–22 October 1997. IEEE.

[JF93] Carsten Lund Joan Feigenbaum, Jeremy A. Kahn. Complexity Results for POMSET Languages. *SIAM Journal on Discrete Mathematics*, 6:432–442, 1993.

[KPR98] Y. Kohayakawa, H. J. Prömel, and Vojtech Rödl. Induced Ramsey Numbers. *Combinatorica*, 18(3):373–404, 1998.

[KT91] Ker-I Ko and Wen-Guey Tzeng. Three Σ_2^p-complete Problems in Computational Learning Theory. *Computational Complexity*, 1:269–310, 1991.

[Lin95] Chih-Long Lin. Optimizing TRIEs for Ordered Pattern Matching is Π_2^p-Complete. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC '95)*, pages 238–245, Los Alamitos, CA, USA, June 1995. IEEE Computer Society Press.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, New York, 1994.

[Ram30] Frank Plumpton Ramsey. On a Problem of Formal Logic. *Proceedings of the London Mathematical Society*, xxx:264–286, 1930.

[Rut86] Vladislav Rutenburg. Complexity of generalized graph coloring. In Jozef Gruska, Branislav Rovan, and Juraj Wiedermann, editors, *Mathematical Foundations of Computer Science, 12th Symposium*, volume 233 of *lncs*, pages 573–581, Bratislava, Czechoslovakia, 25–29 August 1986. Springer.

[SS99] Marcus Schaefer and Daniel Stefankovič. Solvability of Graph Inequalities. Technical Report TR-99-05, Department of Computer Science, University of Chicago, August 1999.

[SS00] Marcus Schaefer and Pradyut Shah. Induced Graph Ramsey Theory. Technical Report TR00-003, DePaul University, June 2000.

[SY80] Yehoshua Sagiv and Mihalis Yannakakis. Equivalences Among Relational Expressions with the Union and Difference Operators. *Journal of the Association for Computing Machinery*, 27(4):633–655, October 1980.

[Uma98] Christopher Umans. The Minimum Equivalent DNF Problem and Shortest Implicants. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 556–563, 1998.

[Uma99] Christopher Umans. Hardness of Approximating Σ_2^p Minimization Problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 465–474, 1999.

[Wag86] Klaus W. Wagner. The Complexity of Combinatorial Problems with Succinct Input Representation. *Acta Informatica*, 23(3):325–356, 1986.