Fundamenta Informaticae 79 (2007) 1–13
IOS Press

A propositional proof system with quantification over permutations

Grzegorz Herman*, Tim Paterson, Michael Soltys
Department of Computing and Software
McMaster University, Ontario, Canada
hermang@mcmaster.ca; paterst@mcmaster.ca; soltys@mcmaster.ca

Abstract. We introduce a new propositional proof system, which we call $\mathbf{H}$, that allows quantification over permutations. In $\mathbf{H}$ we may write $(\exists ab)\alpha$ and $(\forall ab)\alpha$, which is semantically equivalent to $\alpha(a,b) \vee \alpha(b,a)$ and $\alpha(a,b) \wedge \alpha(b,a)$, respectively. We show that $\mathbf{H}$ with cuts restricted to Σ_1 formulas (we denote this system $\mathbf{H_1}$) simulates efficiently the Hajós calculus (**HC**) for constructing graphs which are non-3-colorable. This shows that short proofs over formulas that assert the existence of permutations can capture polynomial time reasoning (as by [9], **HC** is equivalent in strength to **EF**, which in turn captures polytime reasoning). We also show that **EF** simulates efficiently $\mathbf{H_1^*}$, which is $\mathbf{H_1}$ with proofs restricted to being tree-like. In short, we show that $\mathbf{H_1^*} \leq_p \mathbf{EF} \leq_p \mathbf{H_1}$.

1. Introduction

Permutation Frege is a textbook propositional proof system with the extra rule: $\alpha(p_1, p_2, \ldots, p_n) \vdash \alpha(p_{\sigma(1)}, p_{\sigma(2)}, \ldots, p_{\sigma(n)})$, where σ is some permutation. In essence, the *permutation rule* allows a bijective renaming of the variables in a formula. In this paper we introduce a new propositional proof system which permits quantification ($\forall$ and $\exists$) over permutations σ.

It is not known whether permutation Frege is stronger than Frege, and in particular it is not known whether it is as strong as renaming Frege which allows a non-injective renaming of variables (and which is equivalent in power to extended Frege). The strength of permutation Frege remains an open problem despite an intense scrutiny (see [11], where the permutation rule is called the symmetry rule and it is studied in the context of resolution; ando also [2]).

To compare the relative strengths of proof systems, we define the notion of efficient simulation (a notion introduced in the seminal paper [5]). If P_1 and P_2 are proof systems, then P_1 *simulates* P_2 (denoted $P_2 \leq_p P_1$) if there exists a fixed polynomial $p(n)$, such that whenever there is a proof π_1 in

*Address for correspondence: Department of Computing and Software, McMaster University, Ontario, Canada

the proof system P_1 of some tautology τ, then there is a proof π_2 of τ in the proof system P_2 such that $|\pi_2| \leq p(|\pi_1|)$. Two proof systems P_1, P_2 are *equivalent* (denoted $P_1 \equiv_p P_2$) if they simulate each other. (In fact, throughout this paper we can replace the notion of *simulation* by p*-simulation* which adds the extra condition that there exists a polynomial time function f such $\pi_2 = f(\pi_1)$.)

Using this terminology, it is an open question whether permutation Frege is equivalent to Frege, or equivalent to extended Frege, or whether it falls somewhere in between. Of course, showing that it falls strictly in between would automatically show the separation of Frege and extended Frege, which is one of the fundamental problems in theoretical computer science. The strength of permutation Frege is a tantalizing problem since renaming Frege turns out to be equivalent to extended Frege, and permutation and renaming Frege seem to be very closely related.

Furthermore, permutations are a basic algebraic notion and hence it is interesting to design a propositional proof system capable of making assertions about them. In particular, from a complexity-theoretic point of view, graph isomorphism is an **NP** problem for which the certificate is a permutation. In figure 1 the graphs G_1, G_2 are isomorphic via the permutation σ: $1 \mapsto 6, 2 \mapsto 1, 3 \mapsto 2, 4 \mapsto 3, 5 \mapsto 4, 6 \mapsto 5$, but graphs G_1, G_3 (and hence G_2, G_3) are not isomorphic. A proof system for reasoning with quantification over permutations is very well suited for expressing properties of graphs (such as graph isomorphism) in a natural way, and in fact we show that a "tiny" fragment of **H**, which we call $\mathbf{H_1}$, is capable of polytime reasoning, by showing that it simulates the Hajós calculus for constructing graphs which are non-3-colorable.

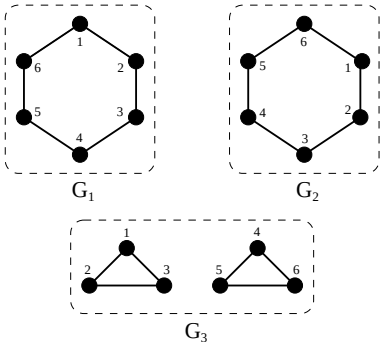

Figure 1. G_1, G_2 are isomorphic, G_1, G_3 are not.

Notice that there is no (known) short certificate of non-isomorphism. An interesting fact about graph isomorphism is that the related **NP** language, i.e., $\{\langle G_1, G_2\rangle |\ G_1, G_2$ are isomorphic graphs $\}$ is one of a few **NP** problems which are neither known known to be in **P** nor to be **NP**-complete. Nevertheless, we show that the related propositional proof system is strong enough to capture polynomial time reasoning.

2. The system PK

We use Gentzen's propositional proof system **PK** (which is equivalent to Frege). For more details on **PK** see [3, 8, 10]. The propositional variables are $p_1, p_2, p_3, \ldots$, but we shall use $a, b, c, \ldots, a_1, b_1, c_1, \ldots$, as meta-variables.

PK is a propositional proof system which operates over sequents. A *sequent* S is written as two sequences of formulas separated by an arrow, i.e., $\alpha_1, \alpha_2, \ldots, \alpha_n \rightarrow \beta_1, \beta_2, \ldots, \beta_m$, where the α_i's and

the β_j's are formulas. The formulas to the left of the arrow are called the *antecedent* and the formulas to the right of the arrow are called the *succedent*. Both are referred to as *cedents*.

A truth assignment τ satisfies a sequent S (written $\tau \models S$), if τ satisfies a formula in Δ or falsifies a formula in Γ. Therefore a sequent S is logically equivalent to the propositional formula $\bigwedge \Gamma \supset \bigvee \Delta$. A sequent is *satisfiable* if it is true under some truth assignment, and *valid* if all truth assignments satisfy it.

A *logical axiom* is a sequent of the form $A \rightarrow A$, where A is any formula. A **PK** *proof* π is a finite sequence of sequents, ending with the sequent that we want to prove: $S_1, S_2, \ldots, S_n$. Each S_i is either an axiom, or follows from one or two previous sequents by a rule. In the **PK** *rules* given in table 1, A and B are formulas, and Γ and Δ are cedents.

Weakening	$\Gamma \rightarrow \Delta \vdash A, \Gamma \rightarrow \Delta$
	$\Gamma \rightarrow \Delta \vdash \Gamma \rightarrow \Delta, A$
Exchange	$\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta \vdash \Gamma_1, B, A, \Gamma_2 \rightarrow \Delta$
	$\Gamma \rightarrow \Delta_1, A, B, \Delta_2 \vdash \Gamma \rightarrow \Delta_1, B, A, \Delta_2$
Contraction	$A, A, \Gamma \rightarrow \Delta \vdash A, \Gamma \rightarrow \Delta$
	$\Gamma \rightarrow \Delta, A, A \vdash \Gamma \rightarrow \Delta, A$
$\vee$	$A, \Gamma \rightarrow \Delta \quad B, \Gamma \rightarrow \Delta \vdash A \vee B, \Gamma \rightarrow \Delta$
	$\Gamma \rightarrow \Delta, A, B \vdash \Gamma \rightarrow \Delta, A \vee B$
$\wedge$	$A, B, \Gamma \rightarrow \Delta \vdash A \wedge B, \Gamma \rightarrow \Delta$
	$\Gamma \rightarrow \Delta, A \quad \Gamma \rightarrow \Delta, B \vdash \Gamma \rightarrow \Delta, A \wedge B$
$\neg$	$\Gamma, A \rightarrow \Delta \vdash \Gamma \rightarrow \neg A, \Delta$
	$\Gamma \rightarrow A, \Delta \vdash \Gamma, \neg A \rightarrow \Delta$
cut	$\Gamma \rightarrow \Delta, A \quad A, \Gamma \rightarrow \Delta \vdash \Gamma \rightarrow \Delta$

Table 1. The rules of PK.

It is well known, and easy to show, that any valid sequent is provable in **PK** (*completeness*), and a sequent with a **PK** proof is valid (*soundness*).

PK is equivalent to Frege, and corresponds to reasoning with $\mathbf{NC}^1$ concepts. This means that the lines of **PK** are sequents of boolean formulas which have the expressive power of $\mathbf{NC}^1$ circuits, which are circuits of polynomial size (in n) and depth $O(\log(n))$, where n is the number of input variables—see [8, 1] for more details.

3. Extensions of PK

Several extensions of **PK** are known which strengthen it to reasoning with polytime concepts. Note that $\mathbf{NC}^1 \subseteq \mathbf{PolyTime}$, and the (conjectured) separation of these two complexity classes is one of the fundamental open problems of theoretical computer science. Note that the lack of a known separation of $\mathbf{NC}^1$ and $\mathbf{PolyTime}$ is mirrored in the lack of a known separation of **PK** with all of the proof systems presented in this section.

$$\frac{\alpha(B), \Gamma \to \Delta}{\forall x \alpha(x), \Gamma \to \Delta} \qquad \frac{\Gamma \to \Delta, \alpha(p)}{\Gamma \to \Delta, \forall x \alpha(x)}$$
$$\frac{\alpha(p), \Gamma \to \Delta}{\exists x \alpha(x), \Gamma \to \Delta} \qquad \frac{\Gamma \to \Delta, \alpha(B)}{\Gamma \to \Delta, \exists x \alpha(x)}$$

Table 2. Rules for introducing quantifiers in **G**.

The most famous extensiuon, equivalent to **Extended Frege** (**EF**), is **Extended PK** (**EPK**) which allows new variables to be introduced and declared to be equivalent to any formula; thus, these new variables serve as abbreviations. Since these abbreviations can later become part of a definition themselves, the effect is that of creating polynomial size circuits, where the abbreviations stand for gates. This is what yields polytime strength of reasoning. The extensions are simply introduced as sequents of the form $\to a \equiv \alpha$ (the connective "$\equiv$" can be simulated with other connectives).

There are many systems equivalent to **EPK**. We now list some of them: **Substitution PK** allows replacing any variable consistently throughout a sequent by a formula; **Renaming PK** allows the renaming of variables, again consistently throughout a sequent, by new variable names; and **T-F PK** allows replacing variables consistently throughout a sequent by the constants T (true) and F (false). See [10, 8] and especially [2] for renaming and **T-F PK**.

Permutation PK is very closely related to **Renaming PK**, but in permutation **PK** there is the added requirement that the renaming has to be bijective. While it is well known that renaming **PK** is equivalent to **EPK**, it is not known whether permutation **PK** is equivalent to **PK** or **EPK** or is strictly in between.

The system **G** (see [8, §4.6] for a complete description) allows boolean quantifiers, and it consists of **PK** together with the four new rules given in table 2. Note that in table 2, B is any formula, and we have the restriction that the atom p does not occur in the bottom sequent for $\forall$-right and $\exists$-left. Semantically,

$$\begin{aligned} \exists x \alpha(x) &\equiv \alpha(0/x) \vee \alpha(1/x) \\ \forall x \alpha(x) &\equiv \alpha(0/x) \wedge \alpha(1/x) \end{aligned} \tag{1}$$

(note that $\alpha(\beta/x)$ means the formula α with every (free) instance of x replaced by β). Thus, quantification does not add to the expressive power of **PK**, but rather allows to shorten formulas. In particular, $\exists x_1 \exists x_2 \ldots \exists x_n \alpha$ would be of length $O(2^n|\alpha|)$ according to the above translation.

We present one more propositional proof system equivalent to **EPK**: the Hajós calculus **HC**, a system for constructing non-k-colorable graphs (so in fact it is a family of proof systems for every $k \geq 3$); we are concerned with $k = 3$, so we shall restrict ourselves to that. Note that **EPK** is a system for deriving valid sequents while **HC** is a system for deriving non-3-colorable graphs; they are equivalent via the standard reductions between satisfiability and 3-colorability. It was shown in [9] that the **HC** is equivalent to **EPK** (the original paper defining the **HC** is [6]). The single axiom of the **HC** is a cliques on 4 vertices,

$K_4 =$ 

and it has three rules:

1. **Addition:** any number of new vertices and edges may be added,

2. **Join:** any two vertices which are not connected may be joined into one, with the resulting duplicate edges removed, and

Figure 2. The join rule: nodes i, j are joined and renamed j.

3. **Edge Elimination:** given two graphs G_1, G_2 on the same vertex set V, and with the edges on all vertices in $V - \{i, j, k\}$ the same, and with (i, j) an edge in both, (i, k) an edge in G_1 only, and (j, k) an edge in G_2 only, we conclude a graph identical to G_1 with the edge (i, k) removed.

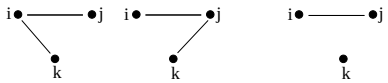

Figure 3. The edge elimination rule. From the two left graphs we conclude the right graph.

As was mentioned above, it was shown in [9] that the **HC** is equivalent to **EPK** (and we use the same presentation of **HC** as in [9]). Interestingly enough, it was shown in [7] that tree-like **HC**, i.e., $\mathbf{HC}^*$ in our notation, has exponential lower bounds (and hence we suspect it to be weaker than dag-like **HC**).

4. PK with permutation quantification (H)

In this section we present a new extension of **PK** which, like **G**, adds rules which introduce quantifiers. However, rather than quantifying over the boolean assignments to a variable (as in **G**), we quantify over *transpositions* of two variables. By combining quantifications over transpositions, we can build quantification over general permutations.

Here is the formula syntax: if α is a formula, then so are $(\exists ab)\alpha$ and $(\forall ab)\alpha$ (note that there are no other restrictions on a and b, so that any, both, or none of them may appear in α and they may be the same variable).

We introduce some notation to give the formal semantics of the new quantifiers. Let $\alpha^{(ab)}$ represent a transposition of the variables a and b within α. That is, every instance of a is replaced by b and every instance of b is replaced by a. For example, $(a \vee b)^{(ab)}$ is $(b \vee a)$. It is important to note that occurrences of a or b within other permutation quantifiers are also affected. Hence, the formula $((\exists ac)\alpha)^{(ab)}$ is syntactically equal to $(\exists bc)\alpha^{(ab)}$.

The intended meaning of permutation quantifiers is as follows:

$$\begin{aligned}(\exists ab)\alpha &\equiv (\alpha \vee \alpha^{(a,b)})\\ (\forall ab)\alpha &\equiv (\alpha \wedge \alpha^{(a,b)})\end{aligned} \tag{2}$$

while for classical boolean quantification the intended meaning is given in (1).

There are two equivalent ways of giving the semantics of permutation quantifiers. Firstly, we can say that the truth assignment τ satisfies a formula $(\exists ab)\alpha$ (written as $\tau \models (\exists ab)\alpha$), iff $\tau \models \alpha$ or $\tau \models \alpha^{(ab)}$. The obvious alternative is to define $\tau^{(ab)}$ by setting $\tau^{(ab)}(a) = \tau(b)$, $\tau^{(ab)}(b) = \tau(a)$, and

$\tau^{(ab)}(x) = \tau(x)$ whenever $x \notin \{a, b\}$. We then say that $\tau \models (\exists ab)\alpha$ iff $\tau \models \alpha$ or $\tau^{(ab)} \models \alpha$. The cases for universal permutation quantifiers are similar, but require both transpositions to be satisfied, rather than just one. The two alternative semantics are equivalent ways of assigning truth values to formulas.

Using several quantifications over transpositions, the existence of a permutation of the variables $p_1, p_2, \ldots, p_n$ can be stated as follows:

$$(\exists p_i p_j)_{1 \leq i,j \leq n} \alpha. \tag{3}$$

Similarly, we can state that α holds for all permutations of a set of variables by replacing $\exists$ with $\forall$ in (3).

Note that this representation is efficient because it requires only n^2 many existential quantifiers (sorting circuits can actually allow an arbitrary permutation to be simulated by only $O(n \log n)$ transpositions). To see that (3) is sufficient to express the existence of an arbitrary permutation in S_n, we have to show that for any $\sigma \in S_n$, the formula $\alpha(p_{\sigma(1)}/p_1, \ldots, p_{\sigma(n)/p_n})$ appears in the long disjunction that we obtain by applying the equivalence given in (2) systematically to get rid of all the transposition quantifiers. A short inductive argument convinces one that this is indeed the case.

The six rules for introducing transposition quantifiers are given in table 3, where α' may be any permutation of those variables in α which occur in the quantifier (for example, if the quantifier is $(\exists ab)$, then α' may be either α or $\alpha^{(ab)}$). The rules R5 and R6 have the following restriction: $a_1, a_2, \ldots, a_n$ do *not* occur or are bound in the rest of the sequent (i.e., every formula in $\Gamma \cup \Delta$ does not contain any a_i, or it starts with an appropriate quantifier over permutation of variables in $S \supseteq \{a_1, \ldots, a_n\}$).

The rules R3 and R4 are required for completeness of $\mathbf{H}$, but not necessary to show $\mathbf{HC} \leq_p \mathbf{H_1}$, while the rules R5 and R6 (in the case of S being strictly equal to $\{a_1, \ldots, a_n\}$) are necessary for showing $\mathbf{HC} \leq_p \mathbf{H_1}$, but not for completeness.

$$\text{R1: } \frac{\alpha, \Gamma \to \Delta}{(\forall ab)\alpha', \Gamma \to \Delta} \qquad \text{R2: } \frac{\Gamma \to \Delta, \alpha}{\Gamma \to \Delta, (\exists ab)\alpha'}$$

$$\text{R3: } \frac{\alpha \vee \alpha^{(ab)}, \Gamma \to \Delta}{(\exists ab)\alpha', \Gamma \to \Delta} \qquad \text{R4: } \frac{\Gamma \to \Delta, \alpha \wedge \alpha^{(ab)}}{\Gamma \to \Delta, (\forall ab)\alpha'}$$

$$\text{R5: } \frac{\alpha, \Gamma \to \Delta}{(\exists a_1 a_2 \ldots a_n)\alpha', \Gamma \to \Delta} \qquad \text{R6: } \frac{\Gamma \to \Delta, \alpha}{\Gamma \to \Delta, (\forall a_1 a_2 \ldots a_n)\alpha'}$$

Table 3. Rules for introducing transposition quantifiers.

Definition 4.1. The system $\mathbf{H}$ consists of the rules of **PK** together with the six new rules in table 3. (We call this system $\mathbf{H}$ to follow the notation introduced in [8], where $\mathbf{G}$ is the name given to the proof system for quantified boolean formulas.)

Theorem 4.1. $\mathbf{H}$ is sound and complete.

Proof:
We know that **PK** is both sound and complete. The soundness of $\mathbf{H}$ follows from the soundness of **PK**, and the soundness of the six new rules in table 3 (which can be easily checked).

We show completeness by induction on the number of permutation quantifiers in a sequent. The base case (no permutation quantifiers) follows from the completeness of **PK**.

In the induction step, in each leaf sequent we pick a formula whose outer-most connective is a permutation quantifier (if there is no such formula we apply the $\neg, \vee, \wedge$ elimination rules until we obtain such a formula). If the quantifier is $\exists$ on the right:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \to \Delta, \alpha, \alpha^{(ab)}}{\Gamma \to \Delta, \alpha, (\exists ab)\alpha}\ \exists\text{-right}}{\Gamma \to \Delta, (\exists ab)\alpha, \alpha}\ \text{exchange-right}}{\Gamma \to \Delta, (\exists ab)\alpha, (\exists ab)\alpha}\ \exists\text{-right}}{\Gamma \to \Delta, (\exists ab)\alpha}\ \text{contraction-right}$$

Note that the top sequent continues being valid if the bottom sequent was valid ($\alpha, \alpha^{(a,b)}$ in a succedent means $\alpha \vee \alpha^{(a,b)}$), and it has one less quantifier, and so we now apply to it the induction hypothesis.

If the quantifier is $\exists$ on the left, we just apply the $\exists$-left rule. The cases for $\forall$ left and right are analogous. Note that we did not need to use rules R5 and R6 to prove completeness. □

Definition 4.2. Let $\Sigma_0 = \Pi_0$ be the set of quantifier-free boolean formulas. Now, recursively, let Σ_{i+1} be the set of all formulas in $\Sigma_i \cup \Pi_i$ plus the formulas of the form $(\exists a_1 a_2 \ldots a_n)\alpha$, where $\alpha \in \Pi_i$, and let Π_{i+1} be defined analogously (with $\forall$ instead of $\exists$). Following the notation of **G** in [8], we let $\mathbf{H_i}$ be **H** with cuts restricted to formulas in Σ_i only, and we let $\mathbf{H_i^*}$ be $\mathbf{H_i}$ restricted to tree-like proofs only (i.e., proofs where each line is used in at most one inference).

Note that unlike in **G**, an arbitrary formula with permutation quantifiers does not necessarily have an equivalent prenex form. On the other hand, the above definition makes no such claims; it only defines a class of formulas, and we are really only interested in $\mathbf{H_1}$.

5. EPK and $\mathbf{H_1}$

In this section we show that **EPK** simulates $\mathbf{H_1^*}$, i.e., $\mathbf{H_1^*} \leq_p \mathbf{EPK}$, and that $\mathbf{H_1}$ simulates the Hajós calculus (**HC**) and hence it simulates **EPK** (i.e., $\mathbf{EPK} \equiv_p \mathbf{HC} \leq_p \mathbf{H_1}$). Since **EPK** corresponds to polytime reasoning, it follows that $\mathbf{H_1}$ captures polytime reasoning (and possibly more—just how much is an open question).

Lemma 5.1. **EPK** simulates $\mathbf{H_1^*}$.

Proof:
This is a witnessing argument, in the style of [8, lemma 4.6.3] where it is shown that **EPK** simulates $\mathbf{G_1^*}$. Roughly speaking, we take an $\mathbf{H_1^*}$ proof, and simulate it line-by-line with a treelike **EPK** proof (**EPK** and treelike **EPK** are ip-equivalent). The novelty is showing how to simulate the six new permutation quantifier rules (given in table 3). In fact, we do not have to deal with the rules introducing universal quantifiers, so we only need to consider R2,R3 and R5, where the last two are taken care off

"automatically", as in the witnessing argument given below the existential quantifiers in the antecedent simply get dropped.

We transform the sequents of the $\mathbf{H}_\mathbf{1}^*$ proof as follows: Σ_0 formulas are unchanged, and for Σ_1 formulas, if we have a sequent with a formula $(\exists p_1 q_1)\cdots(\exists p_n q_n)\alpha$, where $\alpha \in \Sigma_0$, i.e., α is quantifier free, we remove the quantifiers, and add "witnessing" definitions on the right of the sequent. These witnessing definitions compute which transposition ought to be taken. For example, a block of transposition quantifiers $(\exists ab)(\exists cd)(\exists ef)$ can be simulated with variables x_{ab}, x_{cd}, x_{ef}, where x_{ab} is 0 if the transposition (ab) is not taken, and 1 otherwise.

Note that since all the cuts of $\mathbf{H}_\mathbf{1}^*$ are on Σ_1 formulas only, it follows that all quantifiers are existential transposition quantifiers, and they occur only in prenex form (because the conclusion is a Σ_0 sequent, and there is no way of getting rid of a formula other than with a cut rule).

Let $\alpha(\phi/a)$ represent the formula α with every instance of the variable a replaced by ϕ, and let:

$$\{y_1, y_2\}_{ab} := [y_1 \leftrightarrow ((x_{ab} \wedge a) \vee (\neg x_{ab} \wedge b))], [y_2 \leftrightarrow ((x_{ab} \wedge b) \vee (\neg x_{ab} \wedge a))]. \tag{4}$$

Then, a formula $(\exists ab)\alpha$ in the *succedent* of a given sequent can be re-stated as $\alpha(y_1/a, y_2/b)$, and the definition $\{y_1, y_2\}_{ab}$ would be added to the *antecedent*. Repeat this for every transposition quantifier in the block, from the outside in; that is, if the next transposition quantifier was $(\exists cd)$, then we obtain the definitions $\{y_3, y_4\}_{cd}, \{y_1, y_2\}_{ab}$ in the antecedent and the formula $[\alpha(y_1/a, y_2/b)](y_3/c, y_4/d)$ in the succedent. It is important to note that x_{ab}, x_{cd}, y_1, y_2 are *new* variables that "belong" to a particular occurrence of $(\exists ab), (\exists cd)$ and α.

If a Σ_1 formula occurs in the antecedent, we simply drop all the quantifiers.

It remains to show how to translate an entire $\mathbf{H}_\mathbf{1}^*$-proof. We apply the conversion just described to every formula in every sequent. As we go through the proof, from the axioms down to the conclusion, we now have to modify the proof inductively to make sure that it is a valid (treelike) $\mathbf{EPK}$-proof (note that we changed the $\mathbf{H}_\mathbf{1}^*$ proof line-by-line, but the resulting proof is not necessarily a valid $\mathbf{EPK}$ proof; we have to fill-in in between the lines). Consider for example the $\exists$-right rule:

$$\frac{\Gamma \to \Delta, \alpha}{\Gamma \to \Delta, (\exists ab)\alpha}.$$

By the induction hypothesis, the top has already been translated to $\vec{\mathbf{v}}, \Gamma' \to \Delta', \alpha'$, where $\vec{\mathbf{v}}$ denotes the definitions already introduced, and it has a valid $\mathbf{EPK}$-proof. Now consider the bottom sequent. We have to introduce $\{y_1, y_2\}_{ab}$ in the antecedent, which we do with weakening, and we have to replace a, b by y_1, y_2, respectively, in α'. Here is were we make a crucial use of the tree-likeness of $\mathbf{H}_\mathbf{1}^*$ proofs; we replace a, b by y_1, y_2, respectively, in α' and its ancestors throughout the entire proof of $\vec{\mathbf{v}}, \Gamma' \to \Delta', \alpha'$ (which is now an $\mathbf{EPK}$-proof, but our translation preserves the tree-likeness of the original $\mathbf{H}_\mathbf{1}^*$-proof). Finally, we obtain an $\mathbf{EPK}$-proof of

$$\{y_1, y_2\}_{ab}, \vec{\mathbf{v}}, \Gamma' \to \Delta', \alpha'(y_1/a, y_2/b).$$

Consider also a cut rule where all the formulas in Γ, Δ are Σ_0, and the cut formula is $(\exists ab)\beta$, where β is also Σ_0:

$$\frac{\Gamma \to \Delta, (\exists ab)\beta \qquad (\exists ab)\beta, \Gamma \to \Delta}{\Gamma \to \Delta}.$$

By the induction hypothesis we have $\mathbf{EPK}$-proofs of

$$\{y_1, y_2\}_{ab}, \Gamma' \to \Delta', \beta(y_1/a, y_2/b) \quad \text{and} \quad \beta, \Gamma'' \to \Delta''.$$

Before we can apply a cut in the new proof we have to *unify* the two sequents. Thus, we replace a, b in β in the right sequent by y_1, y_2, respectively, throughout the proof of the right sequent, and then do the cut. The same idea works for when the upper sequents are general Σ_1 formulas. The other rules involve similar considerations. The resulting proof has a small growth of size. Since Γ, Δ are Σ_0 formulas, $\Gamma' = \Gamma'' = \Gamma$ and $\Delta' = \Delta'' = \Delta$. In the case that they have Σ_1 formulas we need to do unification on those as well.

Clearly, this is a polytime transformation.

One final observation is that $\mathbf{H}_\mathbf{1}^*$ is a proof system for Σ_0-sequents, i.e., sequents whose formulas are quantifier-free (i.e., the conclusion of an $\mathbf{H}_\mathbf{1}^*$-proof is a Σ_0-sequent, but in the middle of the proof we may have Σ_1-sequents, whose Σ_1-formulas must be eliminated with cuts before reaching the conclusion). At the end of the translation we therefore end up with an $\mathbf{EPK}$-proof of $\vec{\mathbf{v}}, \Gamma \to \Delta$ (note that since Γ, Δ have only Σ_0-formulas, they have not been modified by the translation). But we still have $\vec{\mathbf{v}}$ in the antecedent; to eliminate $\vec{\mathbf{v}}$ we now use the power of $\mathbf{EPK}$: we introduce the abbreviations in $\vec{\mathbf{v}}$ at the *beginning* of the new $\mathbf{EPK}$-proof (to ensure that the abbreviations introduce *new* variables), and once we reach $\vec{\mathbf{v}}, \Gamma \to \Delta$, we use these definitions and the cut rule to eliminate $\vec{\mathbf{v}}$. □

For the other direction, that is, to show that $\mathbf{H_1}$ simulates $\mathbf{EPK}$, we show that it proves efficiently the soundness of the Hajós calculus. This is enough to conclude that $\mathbf{H_1}$ can simulate extended **PK** (see [4, 8]).

To show that $\mathbf{H_1}$ proves efficiently the soundness of the **HC**, it is convenient to use a different formulation of $\mathbf{H_1}$. We define it in terms of Π_1 formulas (i.e., $\forall \pi \alpha$, where $\alpha \in \Sigma_0 = \Pi_0$). Note that the two formulations of $\mathbf{H_1}$ are equivalent, as we can always change sides of all the formulas in the sequent, and obtain an equivalent sequent except all quantifiers have been flipped. It is a matter of expediency which of the two formulations of $\mathbf{H_1}$ we use (as long as we do not mix them together!): for the proof of the simulation of $\mathbf{H}_\mathbf{1}^*$ by extended **PK**, we prefer to use the Σ_1 definition. To show that $\mathbf{H_1}$ proves the soundness of the **HC**, we prefer to use the Π_1 definition.

First, we define a propositional encoding for non-3-colorability of graphs as follows. We fix a vertex set $V = \{1, \ldots, n\}$, with n large enough to contain all the intermediate graphs from the **HC** proof. Every graph will be represented by its edge set E (all vertices "outside" of the graph are kept isolated, which does not influence graph colorability). Our formulas will use variables r_i, g_i, b_i (for $i \in V$), with the intended meaning of vertex i being colored red, green or blue, respectively. We now define:

$$\text{INV} := \bigvee_{i \in V} (r_i \wedge g_i) \vee \bigvee_{i \in V} (r_i \wedge b_i) \vee \bigvee_{i \in V} (g_i \wedge b_i) \vee \bigvee_{i \in V} (\neg r_i \wedge \neg g_i \wedge \neg b_i)$$

and

$$\text{BAD}_E := \bigvee_{\{i,j\} \in E} (r_i \wedge r_j) \vee \bigvee_{\{i,j\} \in E} (g_i \wedge g_j) \vee \bigvee_{\{i,j\} \in E} (b_i \wedge b_j),$$

denoting that the current variable assignment is not valid as a coloring (INV) and that it is a bad coloring with respect to the edge set E (BAD_E), respectively. Finally, we encode non-3-colorability as

$$\text{N3C}_E := (\forall r_1 \ldots r_n)(\forall g_1 \ldots g_n)(\forall b_1 \ldots b_n)(\text{INV} \vee \text{BAD}_E)$$

(note, that even without permutation quantifiers, N3C_E encodes the non-3-colorability of the graph with edge set E; we need the quantifiers however to prove the soundness of the join rule of the **HC**).

Let us also define $E^{(ij)}$ as E with vertices i and j transposed, and $E^{i\to j}$ as E with all the edges incident on i redirected to j.

We are now ready to formally prove the soundness of the **HC** in $\mathbf{H_1}$.

Lemma 5.2. $\mathbf{H_1}$ proves the soundness of the Hajós calculus.

Proof:
It is easy to show that $\mathbf{H_1}$ proves N3C_E whenever E is a 4-clique (we do it without the quantifiers, and then introduce them) because it is a constant size graph. We now analyze each of the inference rules of the **HC**:

1. **Addition:** Having
$$E \subseteq E'$$
we want $\mathbf{H_1}$ to prove
$$\text{N3C}_E \to \text{N3C}_{E'}.$$
We start with the axiom
$$\text{INV} \vee \text{BAD}_E \to \text{INV} \vee \text{BAD}_E,$$
use weakening and exchanges (and the fact that $E \subseteq E'$) to get
$$\text{INV} \vee \text{BAD}_E \to \text{INV} \vee \text{BAD}_{E'},$$
introduce the universal quantifiers (R1) one-by-one on the left
$$\text{N3C}_E \to \text{INV} \vee \text{BAD}_{E'},$$
and finally on the right (R6 – all quantified transpositions are already bound on the left):
$$\text{N3C}_E \to \text{N3C}_{E'}.$$

2. **Join:** Having
$$\{i,j\} \notin E$$
we want $\mathbf{H_1}$ to prove
$$\text{N3C}_E \to \text{N3C}_{E^{i\to j}}.$$
Start with the axiom
$$\text{INV} \vee \text{BAD}_{E^{i\to j}} \to \text{INV} \vee \text{BAD}_{E^{i\to j}}.$$
Use weakenings and exchanges (and the fact that every edge in $E^{i\to j}$ appears in at least one of E or $E^{(ij)}$) to get
$$\text{INV} \vee \text{BAD}_E, \text{INV} \vee \text{BAD}_{E^{(ij)}} \to \text{INV} \vee \text{BAD}_{E^{i\to j}}$$
and then (based on the fact that $\text{BAD}_{E^{(ij)}} \overset{synt.}{=} \text{BAD}_E^{(r_ir_j)(g_ig_j)(b_ib_j)}$) to arrive at
$$(\text{INV} \vee \text{BAD}_E), (\text{INV} \vee \text{BAD}_E)^{(r_ir_j)(g_ig_j)(b_ib_j)} \to \text{INV} \vee \text{BAD}_{E^{i\to j}}.$$

Now use rule R1, exchanges and contraction to get

$$\text{N3C}_E \rightarrow \text{INV} \vee \text{BAD}_{E^{i\rightarrow j}}.$$

Finally, introduce the universal quantifiers on the right (R6 – again, all the transpositions are already bound on the left), and we are done.

3. **Edge Elimination:** Having

$$\begin{aligned} \{i,j\} &\in E, \\ \{i,k\}, \{j,k\} &\notin E, \\ E_1 &= E \cup \{\{i,k\}\}, \\ E_2 &= E \cup \{\{j,k\}\}, \end{aligned}$$

we want $\mathbf{H_1}$ to prove

$$\text{N3C}_{E_1}, \text{N3C}_{E_2} \rightarrow \text{N3C}_E.$$

Consider any pair of edges $\{u,v\} \in E_1$, $\{x,y\} \in E_2$. If $\{u,v\} \neq \{i,k\}$, then $\{u,v\} \in E$ and, by weakening the axiom

$$(r_u \wedge r_v) \vee (g_u \wedge g_v) \vee (b_u \wedge b_v) \rightarrow (r_u \wedge r_v) \vee (g_u \wedge g_v) \vee (b_u \wedge b_v)$$

we can prove

$$((r_u \wedge r_v) \vee (g_u \wedge g_v) \vee (b_u \wedge b_v)) \wedge ((r_x \wedge r_y) \vee (g_x \wedge g_y) \vee (b_x \wedge b_y)) \rightarrow \text{INV} \vee \text{BAD}_E.$$

The case for $\{x,y\} \neq \{j,k\}$ is analogous. If $\{u,v\} = \{i,k\}$ and $\{x,y\} = \{j,k\}$, then we start with

$$\begin{aligned} r_i \wedge r_j &\rightarrow r_i \wedge r_j \\ g_i \wedge g_j &\rightarrow g_i \wedge g_j \\ b_i \wedge b_j &\rightarrow b_i \wedge b_j \\ r_k \wedge g_k &\rightarrow r_k \wedge g_k \\ &\dots, \end{aligned}$$

weaken them to

$$\begin{aligned} (r_i \wedge r_k) \wedge (r_j \wedge r_k) &\rightarrow \text{BAD}_E \\ (g_i \wedge g_k) \wedge (g_j \wedge g_k) &\rightarrow \text{BAD}_E \\ (b_i \wedge b_k) \wedge (b_j \wedge b_k) &\rightarrow \text{BAD}_E \\ (r_i \wedge r_k) \wedge (g_j \wedge g_k) &\rightarrow \text{INV} \\ &\dots, \end{aligned}$$

introduce the connectives, weaken the right size and reorganize the left to get

$$((r_u \wedge r_v) \vee (g_u \wedge g_v) \vee (b_u \wedge b_v)) \wedge ((r_x \wedge r_y) \vee (g_x \wedge g_y) \vee (b_x \wedge b_y)) \rightarrow \text{INV} \vee \text{BAD}_E.$$

Having arrived at the same conclusion for all the pairs of edges from E_1 and E_2, we can combine them together introducing the connectives to get

$$\text{BAD}_{E_1} \wedge \text{BAD}_{E_2} \rightarrow \text{INV} \vee \text{BAD}_E.$$

Reorganizing and weakening the left side slightly gives us

$$\text{INV} \vee \text{BAD}_{E_1}, \text{INV} \vee \text{BAD}_{E_2} \rightarrow \text{INV} \vee \text{BAD}_E,$$

and now we can introduce the quantifiers (as usual, first using R1 and then R6), to get

$$\text{N3C}_{E_1}, \text{N3C}_{E_2} \rightarrow \text{N3C}_E,$$

as required.

Having analyzed all the axioms and inference rules of **HC**, we see that it can be proved sound by $\mathbf{H_1}$. □

Putting together lemmas 5.1 and 5.2, and the fact that **HC** simulates $\mathbf{EPK}$, we arrive at the main result of this work: $\mathbf{EPK}$ simulates $\mathbf{H_1^*}$, and in turn $\mathbf{H_1}$ simulates $\mathbf{EPK}$. This is stated succinctly in theorem 5.1 below.

Theorem 5.1. $\mathbf{H_1^*} \leq_p \mathbf{EPK} \leq_p \mathbf{H_1}$.

6. Conclusion

We have introduced $\mathbf{H}$, a propositional proof system with quantification over permutations. Our system $\mathbf{H}$ does not increase the expressive power of boolean formulas, but allows for abbreviations: $(\exists ab)\alpha \equiv (\alpha \vee \alpha^{(a,b)})$, and $(\forall ab)\alpha \equiv (\alpha \wedge \alpha^{(a,b)})$. It turns out that these abbreviations make a small fragment of the system $\mathbf{H}$ capable of capturing polytime reasoning (we showed, theorem 5.1, that $\mathbf{H_1}$ simulates $\mathbf{EPK}$, and thus Extended Frege). This is interesting because permutations are a fundamental concept of algebra, and we wanted to design a propositional proof system where reasoning about the existence of permutations ($\mathbf{H}$) is built in as a "primitive." Since $\mathbf{G_1^*}$ simulates $\mathbf{EF}$, the obvious next question is: can $\mathbf{H_1^*}$ be shown equivalent to $\mathbf{EF}$? Unfortunately, by [7] tree-like **HC** is not equivalent to dag-like **HC**, and so we cannot claim directly that $\mathbf{H_1^*}$ is equivalent to $\mathbf{EF}$. So we pose the following obvious question: is $\mathbf{H_1^*}$ equivalent to $\mathbf{EF}$?

Acknowledgments. We are very grateful for very valuable comments of anonymous referees that led to substantial improvements of this paper.

References

[1] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. In *Bulletin of the EATCS*, volume 65, pages 66–89. Springer-Verlag, June 1998.

[2] Samuel R. Buss. Some remarks on the lengths of propositional proofs. *Archive for Mathematical Logic*, 34:377–394, 1995.

[3] Samuel R. Buss. An introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, pages 1–78. North Holland, 1998.

[4] Stephen Cook and Phuong Nguyen. Foundations of proof complexity: Bounded arithmetic and propositional translations. Available from `www.cs.toronto.edu/~sacook/csc2429h/book/`, 2006.

[5] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *JSL*, 44:36–50, 1979.

[6] G. Hajós. Über eine konstruktion nicht n-färbbarer graphen. *Wiss. Zeitschr. Martin Luther Univ. Halle-Wittenberg*, 10:116–117, 1961.

[7] Kazuo Iwama and Toniann Pitassi. Exponential lower bounds for the tree-like Hajós calculus. *Inf. Process. Lett.*, 54(5):289–294, 1995.

[8] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge, 1995.

[9] Toniann Pitassi and Alasdair Urquhart. The complexity of the Hajós calculus. *SIAM J. Disc. Math.*, 8(3):464–483, August 1995.

[10] Alasdair Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[11] Alasdair Urquhart. The symmetry rule in propositional logic. *Discrete Applied Mathematics*, 96–97:177–193, 1998.