Propositional PSPACE Reasoning with Boolean Programs vs. Quantified Boolean Formulas*

Alan Skelley**

Department of Computer Science, University of Toronto
10 King's College Road,
Toronto, ON M5S 3G4 Canada
skelley@acm.org

Abstract. We present a new propositional proof system based on a somewhat recent characterization of polynomial space (PSPACE) called Boolean programs, due to Cook and Soltys. The Boolean programs are like generalized extension atoms, providing a parallel to extended Frege. We show that this new system, BPLK, is polynomially equivalent to the system G, which is based on the familiar but very different quantified Boolean formula (QBF) characterization of PSPACE due to Stockmeyer and Meyer. This equivalence is proved by way of two translations, one of which uses an idea reminiscent of the ϵ-terms of Hilbert and Bernays.

1 Introduction

When formulated in a Gentzen sequent style, many known propositional proof systems can be seen to be very similar, with the only difference between them being the computational power of what can be written at each line of the proof (or alternatively, what is allowed in the **cut** rule). Examples are Boolean formulas in Frege systems, single literals in resolution, Boolean circuits in extended Frege systems. Another example is the system G [9], which is a sequent-based system where formulas in the sequents are quantified boolean formulas (QBFs). These formulas have propositional variables and also propositional quantifiers. In this case, then, since evaluating QBFs is PSPACE-complete [12], the computational power that can be harnessed in sequents is PSPACE. We can restrict G to G_i by restricting the number of alternations of quantifiers allowed in the formulas, and the reasoning power is then that of Σ_i^p predicates. G and its subsystems are related to Buss' theories U_2^1, S_2^i and T_2^i [1] by translation and provability of reflection principles [8],[10].

Boolean programs were introduced by Cook and Soltys in [3]. A Boolean program defines a sequence of *function symbols*, where each function symbol is defined using a boolean formula that can include, in addition to the arguments to the function, invocations of the previously defined symbols. The authors of that

* An expanded form [11] of this paper is available from ECCC in the 'Theses' section

** Research supported by Canadian Natural Sciences and Engineering Research Council grant PGSA–208264–1998

paper showed that the problem of evaluating an invocation of a function symbol defined in this way, given the inputs and the Boolean program, is PSPACE complete. The question, then, is whether a proof system formulated around Boolean programs would be equivalent to G. For this to occur, not only would Boolean programs and quantified Boolean formulas need to characterize the same complexity class, but there would need to be an effective way of translating between the two.

This paper answers that question in the affirmative. We define a new system BPLK in a straightforward way to take advantage of the expressive power of Boolean programs, and give translations in both directions between it and G. Although it is possible to restrict Boolean programs to be equivalently expressive as Σ_i^q QBFs (i.e. of a particular number of alternations), we do not address the possibility of matching G_i by a subsystem of BPLK; this would probably be much more technical but not much more interesting. And although BPLK seems convenient to work with in practice, we do not consider issues of its precise relative efficiency to G. It is hoped only that a new and exotic PSPACE proof system to contrast with G may be illuminating.

2 Preliminaries

We assume standard terminology about propositional proof systems and the sequent calculus PK, for example from [7]. Recall that the inference rules are **weakening**, **exchange**, **contraction**, left- and right- introduction rules for each connective, and **cut**. G is the PK with the addition of left- and right- introduction rules for the propositional quantifiers. We adopt the convention of separate sets of free and bound variables as in [2] and call a *semiformula* any formula containing a free occurrence of a bound variable. It should be noted that although G and its subsystems derive tautological statements of quantified propositional logic, in this paper we shall consider them only as proof systems for propositional tautologies.

2.1 Boolean Programs

Boolean programs were introduced in [3] and are a way of specifying Boolean functions. They are something like a generalization of the technique of using new atoms to replace part of a Boolean formula, which idea is the basis of extended Frege systems. The following definition is from that thesis:

Definition 1 (Cook-Soltys). *A* Boolean Program *P is specified by a finite sequence $\{f_1, ..., f_m\}$ of function symbols, where each symbol f_i has an associated arity k_i, and an associated defining equation*

$$f_i(\overline{p_i}) := A_i$$

where $\overline{p_i}$ is a list $p_1, ..., p_{k_i}$ of variables and A_i is a formula all of whose variables are among $\overline{p_i}$ and all of whose function symbols are among $f_1, ..., f_{i-1}$. In *this context the definition of a* formula *is expanded to allow the use of function symbols as connectives.*

The semantics are as for propositional formulas, except that when evaluating an occurrence $f_i(\overline{\phi})$ of a function symbol, the value is defined, using the defining equation, to be $A_i(\overline{\phi})$. There is no free/bound distinction between variables in the language of Boolean programs.

An interesting property of Boolean programs that demonstrates their comparability to quantified Boolean formulas is the following theorem from [3]:

Theorem 2 (Cook-Soltys). *A Language L is in PSPACE iff L is computed by some uniform polynomial size family of Boolean programs.*

2.2 Notational Conventions

We shall use the following conventions of notation: Lower case Latin letters will represent atoms, with $x, y, z, ...$ reserved for bound variables, and with the further exception of $f, g, h, ...$ to be used for function symbols. Capital letters and lower case Greek letters will be used for formulas. An overline indicates a list: $\overline{a}$ is a list of variables $(a_1, ...)$ and $\overline{\overline{A}}$ is a list of lists of formulas ($\overline{A_1} = \{A_{1,1}, A_{1,2}, ...\}, \overline{A_2}, ...$). A formula A may have free variables $\overline{p}$, and when we wish to emphasize that fact we shall write $A(\overline{p})$, although we may not explicitly display all free variables of A. $A(\overline{\phi})$ denotes the result of substituting the list of formulas ϕ for the free variables of A. Since we have separated bound and free variables, in the quantified case we are automatically assured that $\overline{\phi}$ is free for $\overline{p}$ in $A(\overline{p})$, which is to say that no free variables of $\overline{\phi}$ will end up bound by any of A's quantifiers in the substitution.

We shall use the following symbols: '=' and '$\supset$' are not in the language of either system we consider, but we shall use them as abbreviations. $\overline{A} = \overline{B}$ abbreviates $((\neg A_1 \vee B_1) \wedge (\neg B_1 \vee A_1) \wedge ... \wedge (\neg A_k \vee B_k) \wedge (\neg B_k \vee A_k))$. $A \supset B$ abbreviates $\neg A \vee B$. The symbol '$\equiv$' will be used to denote syntactic equality.

3 BPLK and G

In this section we introduce the sequent system BPLK, which is basically PK enhanced with the reasoning power of Boolean programs. We then state two lemmas about BPLK and G that show that certain classes of proofs are easy to find in the two systems, and which will simplify arguments later on. The proofs of these lemmas are technical and have been omitted.

Definition 3 (BPLK). *The system BPLK is like the propositional system PK, but with the following changes:*

1. *In addition to sequents, a proof also includes a Boolean program that defines functions. A BPLK-proof is a pair $\langle \pi, P \rangle$ of the proof (sequents) and the Boolean program defining the function symbols occurring in the sequents.*

2. *Formulas in sequents are formulas in the context of Boolean programs, as defined earlier.*
3. *If the Boolean program contains a definition of the form* $f(\overline{p}) := A(\overline{p})$*, then the new rules*

$$f : \textbf{left} \quad \frac{A(\overline{\phi}), \Gamma \longrightarrow \Delta}{f(\overline{\phi}), \Gamma \longrightarrow \Delta} \qquad \textit{and} \qquad f : \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, A(\overline{\phi})}{\Gamma \longrightarrow \Delta, f(\overline{\phi})}$$

   may be used, where $\overline{\phi}$ *are precisely as many formulas as* $\overline{p}$ *are variables.*
4. *(***Substitution Rule***) The new inference rule* **subst**

$$\frac{\Delta(q, \overline{p}) \longrightarrow \Gamma(q, \overline{p})}{\Delta(\phi, \overline{p}) \longrightarrow \Gamma(\phi, \overline{p})}$$

   may be used, where all occurrences of q *have been substituted for.*

Simultaneous substitutions can be simulated with several applications of **subst**. It is an interesting open problem to eliminate **subst** altogether, perhaps along the lines of the simulation of substitution Frege by extended Frege, as in [4].

Lemma 4 (Propositional Reasoning). *If* $\overline{a}$ *are variables and* $\overline{A}$ *are formulas, and if* $\Gamma'(\overline{a}) \longrightarrow \Delta'(\overline{a})$ *follows from* $\Gamma(\overline{a}) \longrightarrow \Delta(\overline{a})$ *via a proof of size* k*, then a proof of the sequent* $\Gamma'(\overline{A}) \longrightarrow \Delta'(\overline{A})$ *from the sequent* $\Gamma(\overline{A}) \longrightarrow \Delta(\overline{A})$ *can be found in time* $O(k(|\overline{A}|)^3)$ *(*$O(k(|\overline{A}|+|P|)^3)$ *in the case of BPLK, with* P *defining all relevant function symbols), and whose length is thus similarly bounded.*

Lemma 5. *Substitution is a derived rule in* G *(i.e. it can be simulated in* G *by a derivation of size polynomial in the size of the resulting sequent).*

4 BPLK P-Simulates *G*

In this section we give a procedure to translate a sequent in the language of G into a semantically equivalent one in the language of BPLK, along with an accompanying Boolean program. This new Boolean program defines function symbols that occur in the translated sequent, and which replace the quantifiers and bound variables from the original one. We then translate a G-proof line-by-line into the language of BPLK, and we show how to fill in the gaps to form a correct proof in the latter system. Incidentally, this technique could easily be adapted to provide an alternate proof of the PSPACE-completeness of Boolean programs, as the proof in [3] did not involve a reduction from QBFs.

We do not use Boolean programs to compute Skolem functions, which is what first springs to mind, because the same formula would have different translations depending on which side of the sequent it occurred in (because of the semantics of sequents). Instead, we adopt a translation similar to the ϵ-calculus of Hilbert and Bernays [5],[6].

4.1 Special Notation

In the translation in this section, we shall use function symbols whose names are based on formulas of G. Because of the delicate nature of these names, we shall be especially detailed and careful about substitutions and the names of free variables. We shall therefore, in this section only, modify our notation:

When a formula A is displayed as $A(\overline{p})$, we shall mean that $\overline{p}$ are *all* the free variables of A. We shall never use the notation $A(q)$ to mean a substitution, but instead that A has the single free variable q. All substitutions will be denoted with a postfix operator $[B/p]$ which means that the formula B (which may be a single variable) is substituted for p in the formula that precedes the operator. We may write several of these in a row, and the meaning is that they are performed in sequence from left to right. We may also interject free variable lists, as in the example $A(a,b)[B(q,r,s)/b](q,r,s,a)[C/s][D/r]$.

4.2 A Translation from the Language of G to That of BPLK

To aid in translating we introduce the following definition, which gives us a well-defined merging operator that says how to combine several Boolean programs into one, eliminating duplicate definitions:

Definition 6 (Merging $P \diamond Q$ of Boolean Programs). *If P and Q are Boolean programs, or at least fragments (not necessarily defining all function symbols used in definitions), then the merging $P \diamond Q$ of P and Q is obtained by first concatenating P and Q, and then deleting all but the first definition of each function symbol.*

We shall use this merging operator in the following translation, and later on in the actual simulation. However, whenever we merge two Boolean programs that both define a particular function symbol, it will always be the case that the two definitions are identical. Thus, which of the definitions is kept is immaterial.

Now we can present the actual translation, defined first for single formulas:

Definition 7 (Translation $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ of ϕ). *We recursively define a translation of a quantified propositional* semiformula *into a semantically equivalent quantifier-free* formula *in the language of BPLK, together with a Boolean program defining the function symbols in that formula.*

- *If ϕ is an atom p then $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ is $\langle \phi, \emptyset\rangle$.*
- *If ϕ is $\psi \wedge \theta$ ($\psi \vee \theta$) then $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ is $\langle \ulcorner\psi\urcorner \wedge \ulcorner\theta\urcorner, P[\psi] \diamond P[\theta]\rangle$ ($\langle \ulcorner\psi\urcorner \vee \ulcorner\theta\urcorner, P[\psi] \diamond P[\theta]\rangle$).*
- *If ϕ is $\neg\psi$ then $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ is $\langle \neg\ulcorner\psi\urcorner, P[\psi]\rangle$.*
- *If ϕ is $\exists x \psi(x,\overline{p})$, then $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ is $\langle f_\phi(\overline{p}), P[\psi] \diamond P'[\phi]\rangle$ Here, $P'[\phi]$ is a Boolean program fragment with the following two definitions:*

$$\epsilon_\phi(\overline{p}) := \ulcorner\psi\urcorner[1/x](\overline{p}) \quad \textit{and} \quad f_\phi(\overline{p}) := \ulcorner\psi\urcorner[\epsilon_\phi(\overline{p})/x](\overline{p}).$$

– *If ϕ is $\forall x\psi(x,\overline{p})$, then $\langle \ulcorner\phi\urcorner, P[\phi]\rangle$ is $\langle f_\phi(\overline{p}), P[\psi] \diamond P'[\phi]\rangle$ Here, $P'[\phi]$ is a Boolean program fragment with the following two definitions:*

$$\epsilon_\phi(\overline{p}) := \ulcorner\psi\urcorner[0/x](\overline{p}) \quad \textit{and} \quad f_\phi(\overline{p}) := \ulcorner\psi\urcorner[\epsilon_\phi(\overline{p})/x](\overline{p}).$$

Thus, in the case where a quantifier is the top-level connective of $\phi = Qx\psi(x,\overline{p})$, we first pick a function symbol whose name depends on the formula ϕ (i.e., including more deeply nested quantifiers and free variables). This choice encodes both which variable is the most newly quantified one, and also what kind of quantifier it is. We call this function symbol a witnessing function for ψ because its value will satisfy ψ if possible, in case Q is existential, or else it will falsify ψ if possible.

We then substitute this function symbol, applied to all the variables free in ϕ, into the *translation* of ψ. Finally, we define a new function symbol using the resulting formula, and use it instead of the formula. We do so because $\ulcorner\psi\urcorner[\epsilon_\phi(\overline{p})/x](\overline{p})$ has more occurrences of free variables than $\ulcorner\psi\urcorner(\overline{p})$, and so without the new function symbol, subsequent substitutions of this form may increase the length of the formula exponentially.

Note that if ϕ is quantifier-free, then it is unchanged by the translation and the Boolean program that results is empty.

Definition 8. *If S is the sequent $\Gamma_1, ..., \Gamma_j \longrightarrow \Delta_1, ..., \Delta_k$ then $\langle \ulcorner S\urcorner, P[S]\rangle$ is*

$$\langle \ulcorner\Gamma_1\urcorner, ..., \ulcorner\Gamma_j\urcorner \longrightarrow \ulcorner\Delta_1\urcorner, ..., \ulcorner\Delta_k\urcorner, \qquad P[\Gamma_1] \diamond ... \diamond P[\Delta_k]\rangle.$$

We call $P[\phi]$ or $P[S]$ the Boolean program *arising* from the translation. As a final lemma in this subsection, we observe that translations are polynomial size:

Lemma 9. *If S is a sequent in the language of G, then $|\langle \ulcorner S\urcorner, P(S)\rangle| \in O(|S|^3)$.*

4.3 A Simulation of G by BPLK

First, we observe that the translations defined above are semantically equivalent to the original formulas.

Lemma 10. *In the presence of the Boolean program $P[\phi]$ defining the function symbols as above (arising from the translation), a quantified Boolean formula ϕ is semantically equivalent to its translation $\ulcorner\phi\urcorner$.*

Now, the operations of substitution and translation do not commute directly even in the case of substituting only a single variable, so for example if $A(a) \equiv \exists x(a \vee x)$, then its translation is $\ulcorner A\urcorner(a) \equiv f_{\exists x(a\vee x)}(a)$. If we then substitute b for a we obtain $\ulcorner A\urcorner[b/a] \equiv f_{\exists x(a\vee x)}(b)$. On the other hand, if we did these things in the reverse order we would get $\ulcorner A[b/a]\urcorner \equiv f_{\exists x(b\vee x)}(b)$, which is different. A technical lemma is needed to resolve this difficulty:

Lemma 11. *Let $A(s, \overline{p}, \overline{r})$ be a semiformula and $B(\overline{p}, \overline{q})$ a formula, both in the language of G, so B is automatically free for s in A, and let $\overline{p}$ be all free variables except s that are common to A and B. (s may be in $\overline{q}$) Then proofs of*

$$\ulcorner A[B/s] \urcorner(\overline{p}, \overline{q}, \overline{r}) \longrightarrow \ulcorner A \urcorner[\ulcorner B \urcorner / s](\overline{p}, \overline{q}, \overline{r})$$

and

$$\ulcorner A \urcorner[\ulcorner B \urcorner / s](\overline{p}, \overline{q}, \overline{r}) \longrightarrow \ulcorner A[B/s] \urcorner(\overline{p}, \overline{q}, \overline{r})$$

can be found in time polynomial in the size of translations and the size of the Boolean program arising from them.

The (quite technical) proof of this lemma is by induction on the structure of A and has been omitted. The main result of the section follows:

Theorem 12. *If S is a quantifier-free sequent with a proof π_1 in G, then S has a BPLK-proof $\langle \pi_2, P[\pi_1] \rangle$ that, given π_1, can be found in time polynomial in $|\pi_1|$ (and thus has polynomial size).*

Proof. First of all, $P[\pi_1]$ is formed by merging the Boolean programs arising from translating all the sequents in π_1. More precisely, if π_1 is $S_1, ..., S_k$, then $P[\pi_1] = P[S_1] \diamond ... \diamond P[S_k]$.

We then form π_2 directly by translating π_1 sequent-by-sequent into the language of BPLK, and when necessary adding some extra sequents between the translated ones. We have the following cases, depending on the inference rule used:

Observe that all initial sequents of G are their own translations, and are also initial sequents of BPLK.

If S is non-initial and is inferred by *any* rule except a quantifier introduction with hypothesis(es) T (and U), then $\ulcorner S \urcorner$ follows from $\ulcorner T \urcorner$ (and $\ulcorner U \urcorner$) by the same rule. This is because translations of identical formulas are identical, and the translation operator commutes with the connectives $\vee$, $\wedge$ and $\neg$.

If S is inferred from T by the introduction of a universal quantifier on the right, or that of an existential quantifier on the left, then considering the first case we have that T is $\Gamma \longrightarrow \Delta, C(a, \overline{p})$ and S is $\Gamma \longrightarrow \Delta, \forall x (C[x/a](x, \overline{p}))$. Their translations are thus $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, \ulcorner C \urcorner(a, \overline{p})$ and $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, f_{\forall x C[x/a]}(\overline{p})$, respectively. For convenience, let $A \equiv C[x/a]$. In this notation, $\ulcorner T \urcorner$ is $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, \ulcorner A[a/x] \urcorner(a, \overline{p})$. First, $\ulcorner A[a/x] \urcorner \longrightarrow \ulcorner A \urcorner[\ulcorner a \urcorner / x]$ follows from lemma 11, taking $B \equiv a$, and $s \equiv x$. $\ulcorner T \urcorner$ and **cut** then yield $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, \ulcorner A \urcorner[a/x]$. Next, observe that a cannot occur in Γ or Δ (or therefore their translations) due to the restriction on $\forall$: **right**. So, when we apply **subst** to this sequent, substituting $\epsilon_{\forall x A}(\overline{p})$ for a to obtain $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, \ulcorner A \urcorner[\epsilon_{\forall x A}(\overline{p})/x]$, $\ulcorner \Gamma \urcorner$ and $\ulcorner \Delta \urcorner$ are unchanged. Finally, $\ulcorner A \urcorner[\epsilon_{\forall x A}(\overline{p})/x] \longrightarrow f_{\forall x A}(\overline{p})$ follows from the definition of the function symbol $f_{\forall x A}$ using the introduction **right** rule for that symbol. $\ulcorner S \urcorner$ follows with **weakening** and **cut**.

The interesting case is when S follows from T by the introduction of an existential quantifier on the right, or that of a universal quantifier on the left. The two cases are symmetrical, so consider the first. T is $\Gamma \longrightarrow \Delta, A(x, \overline{p})[B/x]$ and S is

$\Gamma \longrightarrow \Delta, \exists x(A(x,\overline{p}))$. The translations $\ulcorner T \urcorner$ and $\ulcorner S \urcorner$ are $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, \ulcorner A[B/x] \urcorner$ and $\ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner, f_{\exists x A}(\overline{p})$. Thus, it suffices to prove $\ulcorner A[B/x] \urcorner \longrightarrow f_{\exists x A}(\overline{p})$ and apply **cut**. First, we derive $\ulcorner A[B/x] \urcorner \longrightarrow \ulcorner A \urcorner [\ulcorner B \urcorner /x]$ using lemma 11. The next task is to prove $\ulcorner A \urcorner [\ulcorner B \urcorner /x] \longrightarrow \ulcorner A \urcorner [\epsilon_{\exists x A}(\overline{p})/x]$. This sequent is proved by reasoning by cases about the value of $\epsilon_{\exists x A}(\overline{p})$ but this proof is omitted for brevity.

Lastly, $\ulcorner A \urcorner [\epsilon_{\exists x A}(\overline{p})/x] \longrightarrow f_{\exists x A}(\overline{p})$ follows from the definition of $f_{\forall x A}$.

Now, starting with $\ulcorner T \urcorner$ and using each of these last three sequents in turn with **weakening** and **cut**, one obtains the sequent $\ulcorner S \urcorner$, as desired.

Finally, in the case that S is quantifier-free (for example, the final sequent in a proof), then the translation of S is S, and this completes the proof of the theorem. $\square$

5 *G* P-Simulates BPLK

In this section we define a translation of Boolean programs into formulas in the language of G, that is, using propositional quantifiers, inductively on the length of the program. To avoid exponential blowup of the translations, we must be careful to use exactly one previous translation at each step, so our construction produces one formula simultaneously translating all the function symbols.

The method we shall use to "multiplex" is inspired by a trick used by Stockmeyer and Meyer in [12] to show that the problem of evaluating a given quantified Boolean formula (QBF) is PSPACE-complete. The idea is to abbreviate $\phi(x_1, y_1) \wedge \phi(x_2, y_2)$ as $\forall x, y[((x = x_1 \wedge y = y_1) \vee (x = x_2 \wedge y = y_2)) \supset \phi(x, y)]$. This is not exactly what is needed but is quite close.

5.1 A Translation from the Language of BPLK to That of *G*

Reserve $\overline{v}$, $\overline{u}$, $\overline{z}$ of the bound type and $\overline{t}$, $\overline{d}$ of the free type as new variables not occurring in the BPLK-proof being translated. There will be variables v_A and t_A for each formula A in the language of BPLK, and we shall replace occurrences of function symbols by these two classes of variables as part of our translation using the two operators defined below. One is used to translate the Boolean program, and the other to translate the sequents of the BPLK proof.

Definition 13 (Hat and Corner Operators). *If $A(\overline{p})$ is a formula in the language of BPLK ($\overline{p}$ are the variables free in A), then let $\widehat{A}(\overline{p}, \overline{v})$ be the result of replacing, in A, every maximal subformula $f(\overline{B})$ of A whose main connective is a function symbol by the corresponding variable $v_{f(\overline{B})}$. This new formula will have some subset of the original $\overline{p}$ and also some v's as free variables. The corner operator (e.g. $\ulcorner A \urcorner$) is defined similarly, except that corresponding t variables are used instead of v's. When either operator is applied to a set of formulas, we mean that it should be applied to each formula in the set in turn.*

Example 14 (Hat Operator). If $\psi(\overline{p})$ is the formula $f(p_1) \vee g(f(p_1 \wedge p_2)) \wedge \neg p_3$, then $\widehat{\psi}(\overline{p}, \overline{v})$ is $v_{f(p_1)} \vee v_{g(f(p_1 \wedge p_2))} \wedge \neg p_3$.

Let us fix a Boolean program defining functions $f_1, f_2, \ldots$ such that each function symbol in the Boolean program is defined as $f_i(\overline{p}) := A_i(\overline{p})$. (In other words, A_i is the defining formula of f_i). Now, $\phi_k(\overline{\overline{z}}, \overline{u})$ will be the translation of the Boolean program up to and including the definition of f_k. The meaning of $\phi_k(\overline{z_1}, \ldots, \overline{z_k}, u_1, \ldots, u_k)$ will be that this formula is satisfied if and only if $f_1(\overline{z_1}) = u_1, \ldots,$ and $f_k(\overline{z_k}) = u_k$.

Definition 15. *Consider a Boolean program defining $f_1 \ldots f_k$. Define $\phi_0 := 1$. Now, say the definition of f_i includes the function symbol occurrences $f_{j_1}(\overline{B_1})$, ..., $f_{j_m}(\overline{B_m})$ in order, some of which may be nested in others. (Here $j_1, j_2, \ldots, j_m$ are just the indexes of the function symbols, i.e., a sequence of m integers, not necessarily distinct, each smaller than i). Then define*

$$
\begin{array}{l}
\phi_i(\overline{z_1}, \ldots, \overline{z_i}, u_1, \ldots, u_i) := \\
\quad \exists \overline{v_{f_j(B_j(\overline{z_i}))}} [\widehat{A_i}(\overline{z_i}, \overline{v_{f_j(B_j(\overline{z_i}))}}) = u_i \wedge \\
\quad\quad \forall \overline{z'_1}, \ldots, \overline{z'_{i-1}}, u'_1, \ldots, u'_{i-1} [\phi_{i-1}(\overline{\overline{z'}}, \overline{u'}) \supset [\\
\quad\quad\quad \left.\begin{array}{l}
(\overline{z'_{j_1}} = \overline{\widehat{B_1(\overline{z_i})}} \supset u'_{j_1} = v_{f_{j_1}(B_1(\overline{z_i}))}) \wedge \\
(\overline{z'_{j_2}} = \overline{\widehat{B_2(\overline{z_i})}} \supset u'_{j_2} = v_{f_{j_2}(B_2(\overline{z_i}))}) \wedge \\
\vdots \\
(\overline{z'_{j_m}} = \overline{\widehat{B_m(\overline{z_i})}} \supset u'_{j_m} = v_{f_{j_m}(B_m(\overline{z_i}))}) \wedge
\end{array}\right\} (*) \\
\quad\quad\quad \left.\begin{array}{l}
(\overline{z'_1} = \overline{z_1} \supset u'_1 = u_1) \wedge \\
\vdots \\
(\overline{z'_{i-1}} = \overline{z_{i-1}} \supset u'_{i-1} = u_{i-1})
\end{array}\right\} (**) \\
\quad\quad\quad] \\
\quad\quad] \\
\quad].
\end{array}
$$

Lemma 16. *For each i, $\phi_i(\overline{\overline{z}}, \overline{u})$ is semantically equivalent to $f_1(\overline{z_1}) = u_1 \wedge \ldots \wedge f_i(\overline{z_i}) = u_i$. (The part $(*)$ above recursively uses ϕ_{i-1} to help evaluate the ith function symbol, and $(**)$ passes through the evaluation of the previous ones.)*

We can now define the translation of sequents. This translation is in the context of a BPLK-proof, so the Boolean program and the rest of the sequents in the proof are already fixed. Exactly which proof a particular translation is relative to is not indicated in the notation, but it will always be clear from the context.

Definition 17 (Translation $\ulcorner S \urcorner$ of the sequent S relative to π). *Fix a BPLK-proof π and its associated Boolean program defining $f_1, \ldots, f_k$. Let $f_{j_i}(\overline{C_i})$ be a list of all subformulas in π whose main connective is a function symbol. ($\overline{C_i}$ are arguments to f_{j_i}, and again j_i are simply indexes).*

Then if S is the sequent $\Gamma \longrightarrow \Delta$, it is translated as the sequent $\ulcorner S \urcorner$:

$$
\begin{array}{l}
\phi_k(0, \ldots, 0, \overline{\ulcorner C_1 \urcorner}, 0, \ldots, 0, d^1_1, \ldots, d^1_{j_1-1}, t_{f_{j_1}(\overline{C_1})}, d^1_{j_1+1}, \ldots, d^1_k), \ldots \\
\phi_k(0, \ldots, 0, \overline{\ulcorner C_m \urcorner}, 0, \ldots, 0, d^m_1, \ldots, d^m_{j_m-1}, t_{f_{j_m}(\overline{C_m})}, d^m_{j_m+1}, \ldots, d^m_k), \ulcorner \Gamma \urcorner \longrightarrow \ulcorner \Delta \urcorner.
\end{array}
$$

Here the $\overline{\ulcorner C_i \urcorner}$ *and the corresponding* t*'s are in the correct places to be the arguments to, and the values of, the function symbol* f_{j_i}*. The* d *are dummy variables. We could use* $t_{f_i(\overline{0})}$ *instead of* $d_i^!$ *(since* $d_i^!$ *will be constrained to the value* $f_i(\overline{0})$*) but it will be convenient later on that the* d*'s are distinct. We shall call the occurrences of* ϕ_k *above the* prefix *of the translation, and the remainder the* suffix.

Now, these translations may have free variables that the original ones did not (t's and d's). We cannot, therefore, assert semantic equivalence of the two. However, we are concerned with proving valid sequents, and we can say something nearly as good:

The idea is that if the translation of a sequent is satisfied by some assignment, then either one of the t or d variables has an incorrect value, falsifying the corresponding instance of ϕ_k, or else they all have the correct values and the remainder of the translated sequent is satisfied. In that case, the original sequent is satisfied by the same assignment. Conversely, if the original sequent is valid, then every assignment to the translation will either falsify one of the ϕ_k's, or else all the t's will have the correct value and thus the remainder of the sequent will be satisfied. Therefore,

Claim. For any sequent S from the language of BPLK, S is valid if and only $\ulcorner S \urcorner$ is.

Lemma 18. *Let* S *be any sequent from the BPLK-proof* $\langle \pi, P \rangle$*. Then* $|\ulcorner S \urcorner| \in O(|P|^2|\pi|^2)$.

5.2 A Simulation of BPLK by *G*

We first show that proofs of sequents from two special classes are efficient to find.

Lemma 19 (Existence and Uniqueness Sequents). *The* existence sequents E_i*:*

$$\longrightarrow \forall \overline{\overline{z}} \exists \overline{u} \phi_i(\overline{\overline{z}}, \overline{u})$$

and uniqueness sequents U_i

$$\begin{array}{l} \longrightarrow \forall \overline{\overline{z_1}}, \overline{\overline{z_2}}, \overline{u_1}, \overline{u_2} [\phi_i(\overline{\overline{z_1}}, \overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}}, \overline{u_2}) \supset [\\ \qquad (\overline{z_{1,1}} = \overline{z_{2,1}} \supset u_{1,1} = u_{2,1}) \wedge \\ \qquad \vdots \\ \qquad (\overline{z_{1,i}} = \overline{z_{2,i}} \supset u_{1,i} = u_{2,i}) \\ \qquad] \\ \quad] \end{array}$$

have proofs that can be found in polynomial time, and thus are of polynomial size.

The two parts of this lemma are proved by induction in parallel. One more lemma, analogous to lemma 11 of the previous section, is needed because substitution does not commute with translation. The proofs of these lemmas are omitted.

Lemma 20. *If $T(p)$ is a sequent in a BPLK-proof and ψ is a BPLK formula in which p does not occur, then a G-proof of $\ulcorner T(\psi)\urcorner$ from $\ulcorner T\urcorner(\ulcorner\psi\urcorner)$ can be found in time polynomial in the size of its endsequent.*

Finally, we can state and prove the main result:

Theorem 21. *If S has a BPLK-proof π_1, then S has a G-proof π_2 that, given π_1, can be found in time polynomial in $|\pi_1|$ (and thus has polynomial size).*

Proof. We construct π_2 directly by translating π_1, sequent-by-sequent, into the language of G, relative to the Boolean program of π_1. If necessary, we insert sequents to prove the translation of a sequent from the translations of its hypotheses.

First of all, if S is an initial sequent of BPLK, then it is function symbol free and so its translation is itself, and thus already an initial sequent of G.

Now consider a non-initial sequent S inferred from previous ones. If the inference was **weakening**, **contraction**, **cut**, or introduction of $\neg$, $\wedge$ or $\vee$, then the same rule yields $\ulcorner S\urcorner$.

If $S = T(\psi)$ is inferred from $T(p)$ by **subst**, then note that without loss of generality we may assume that p does not occur in ψ. Otherwise we could modify π_1 to perform **subst** twice; once to substitute $\psi(q)$ for p (q is a variable that does not occur in T) and then again to substitute p for q. To simulate the substitution in G, first use lemma 5 to substitute $\ulcorner\psi\urcorner$ for p in $\ulcorner T\urcorner(p,\overline{t})$, obtaining $\ulcorner T\urcorner(\ulcorner\psi\urcorner,\overline{t})$. Finally, apply lemma 20.

The last case in the proof is when S is inferred by f_i-introduction, introducing $f_i(\overline{B(\overline{p})})$. Then clearly the sequent $\phi_k(...),... \longrightarrow \ulcorner A_i(\overline{B})\urcorner(\overline{p},\overline{t}) = t_{f_i(\overline{B})}$, together with some propositional reasoning, will produce $\ulcorner S\urcorner$ (basically just by using **cut**). We derive the desired sequent as follows: First, the following is straightforward:

$$\phi_k(\overline{0},\overline{\ulcorner B\urcorner},\overline{u},t_{f_i(\overline{B})},\overline{d}) \longrightarrow \phi_i(\overline{0},\overline{\ulcorner B\urcorner},\overline{u},t_{f_i(\overline{B})},\overline{d}).$$

Next, we add the rest of the prefix:

$$\phi_k,... \longrightarrow \phi_i(\overline{0},\overline{\ulcorner B\urcorner},\overline{u},t_{f_i(\overline{B})},\overline{d}).$$

Expanding the ϕ_i,

$$\phi_k,... \longrightarrow \exists\overline{v}[\widehat{A_i(\overline{z_i})}(\overline{\ulcorner B\urcorner},\overline{v}) = t_{f_i(\overline{B})} \wedge ....].$$

Note that the A_i occurrence above contains t variables, from the $\overline{\ulcorner B\urcorner}$ substituted for the $\overline{z_i}$, and also v variables, from function symbols occurring in the definition of f_i. Uniqueness for ϕ_{i-1} and

$$\phi_k(\overline{0},\overline{\ulcorner C(\overline{B})\urcorner},\overline{u},t_{f_j(\overline{C(\overline{B})})},\overline{d}) \longrightarrow \phi_{i-1}(\overline{0},\overline{\ulcorner C(\overline{B})\urcorner},\overline{u},t_{f_j(\overline{C(\overline{B})})},\overline{d}),$$

one sequent for each function symbol occurrence $f_j(\overline{C((\overline{z_i}))})$ in the definition of f_i, allow us to rename the $v_{f_j(\overline{C(\overline{z_i})})}$ in the occurrence of $\widehat{A_i}$ above to $t_{f_j(\overline{C(\overline{B}))}}$, producing $\ulcorner A_i(\overline{B})\urcorner$, and then we drop the existential quantifier and some conjuncts to get $\phi_k(...), ... \longrightarrow \ulcorner A_i(\overline{B})\urcorner = t_{f_i(\overline{B})}$, which is the desired sequent.

Nearing the end of the proof now, if S is the last sequent of the proof, then it is function symbol-free. We need only remove the prefix from $\ulcorner S \urcorner$ to obtain S. The t variable corresponding to the outer-most function symbol occurrence in π_1 (there may be many outer-most occurrences) is defined by an occurrence of ϕ_k, but it is not used in the definition of any of the other t variables. We may thus use $\exists$: **left** on the t and the d's, followed by $\forall$: **left** on the B's and the 0's, to change this occurrence into $\forall\overline{\overline{z}}\exists\overline{u}\phi_k(\overline{\overline{z}},\overline{u})$, which we can cut away with the existence sequent and **weakening**. We can now do the same for the next most outer function symbol occurrence, and so on. The resulting sequent at the end of this process is S, which completes the proof. ☐

Acknowledgment

The author wishes to thank the anonymous referees and, in the strongest possible terms, his supervisor, Stephen Cook, for countless helpful discussions.

References

[1] S. Buss. *Bounded Arithmetic.* Bibliopolis, Naples, 1986.
[2] Samuel R. Buss, editor. *Handbook of Proof Theory.* Elsevier Science B. V., Amsterdam, 1998.
[3] Stephen Cook and Michael Soltys. Boolean programs and quantified propositional proof systems. *Bulletin of the Section of Logic*, 28(3), 1999.
[4] Martin Dowd. Model theoretic aspects of P $\neq$ NP. Typewritten manuscript, 1985.
[5] D. Hilbert and P. Bernays. *Grundlagen der Mathematik I.* Springer, Berlin, 1934.
[6] D. Hilbert and P. Bernays. *Grundlagen der Mathematik II.* Springer, Berlin, 1939.
[7] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.
[8] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.
[9] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschr. f. Mathematikal Logik u. Grundlagen d. Mathematik*, 36:29–46, 1990.
[10] Jan Krajíček and Gaisi Takeuti. On bounded Σ_1^1 polynomial induction. In S. R. Buss and P. J. Scott, editors, *FEASMATH: Feasible Mathematics: A Mathematical Sciences Institute Workshop*, pages 259–80. Birkhauser, 1990.
[11] Alan Skelley. Relating the PSPACE reasoning power of Boolean programs and quantified Boolean formulas. Master's thesis, University of Toronto, 2000. Available from ECCC in the 'theses' section.
[12] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Conference Record of Fifth Annual ACM Symposium on Theory of Computing*, pages 1–9, Austin, Texas, 30 April–2 May 1973.

A Appendix

A.1 Proof of Lemma 4

First another lemma:

Lemma 22. *If ϕ is any formula in the language of either BPLK or of G, then the sequent*

$$\phi \longrightarrow \phi$$

has a proof (in the appropriate system) that can be found in time polynomial in $|\phi|$ (and in the size of the Boolean program defining the function symbols in ϕ, as appropriate) and is therefore polynomial size.

Proof. The proof is a simple induction on the structure of ϕ:

- If ϕ is a variable then the result is immediate.
- If ϕ is $\neg\psi$, then to the sequent

  $$\psi \longrightarrow \psi$$

  apply $\neg$: **right** and $\neg$: **left**.
- If ϕ is $\psi \vee \theta$ then apply **weakening** then $\vee$: **right** to each of the sequents

  $$\psi \longrightarrow \psi$$

  and

  $$\theta \longrightarrow \theta,$$

  then apply $\vee$: **left**.
- The case for $\wedge$ is symmetric.
- If ϕ is $Qx\psi(x)$ for some quantifier Q then to the sequent

  $$\psi(a) \longrightarrow \psi(a)$$

  apply both introduction rules for the quantifier, left first for universal, right first for existential.
- If ϕ is $f(\overline{\psi})$, then by a separate induction on i, derive

  $$f_i(p) \longrightarrow f_i(p),$$

  using **subst** and the other cases of the current lemma to derive

  $$A_i(\overline{p}) \longrightarrow A_i(\overline{p}),$$

  and then applying f_i-introduction. Finally, apply **subst** to $f(\overline{p}) \longrightarrow f(\overline{p})$.

We apply at most 5 inferences for each connective in ϕ (and P), and the proof can easily be computed using a recursion on the structure of ϕ, so $r(x) = x^2$ is sufficient. □

Proof (Proof of lemma 4). Simply substitute $\overline{A}$ for $\overline{a}$ in the original proof. The only case to consider is if a sequent in the original proof is an initial sequent, in which case we can apply lemma 22. The substitution expands the original proof by at most a factor of $|\overline{A}|$ and adding the subproofs obtained from lemma 22 adds a factor of at most $5|\overline{A}|^2$ so $r(k, |\overline{A}|) = k(|\overline{A}|)^3$ is sufficient. □

A.2 Proof of Lemma 5

Proof. The proof is as follows: First apply $\neg$: **right** then $\vee$: **right** to obtain an equivalent sequent with a single formula,

$$\longrightarrow F_S(\overline{p}).$$

Apply $\forall$: **right** to obtain

$$\longrightarrow \forall \overline{x} F_S(\overline{x}).$$

By lemma 22, the sequent

$$F_S(\overline{\phi}) \longrightarrow F_S(\overline{\phi})$$

has a short proof, and thus $\forall$: **left**, **weakening** and **cut** produce

$$\longrightarrow F_S(\overline{\phi}).$$

The desired sequent follows after some propositional reasoning. □

A.3 Proof of Lemma 9

Proof. First, note that $|\ulcorner \phi \urcorner| \in O(|\phi|)$: Inductively following the recursive definition, all the Boolean connective cases increase the translation size linearly. In the case of $\phi \equiv \exists x \psi(x, \overline{p})$, the translation $\ulcorner \phi \urcorner$ is just $f_\phi(\overline{p})$, so the number of symbols in the translation is at most twice the number in the original formula. This is because to write the translation we write just an f, a copy of ϕ, and then a list of ϕ's free variables.

Now, the Boolean program arising from a translation of ϕ contains at most two function symbols for each quantifier in ϕ. The size of the names of the function symbols is linear in $|\phi|$, since the name is based on a subformula of ϕ. The defining formula for an ϵ function symbol is linear in size from the previous paragraph, since it is essentially just a translation of a subformula of ϕ. The defining formula for an f function symbol consists of a translation of a subformula of ϕ, of linear size, with an ϵ function symbol substituted for one of the variables, and therefore is at most quadratic in size. The size of the entire Boolean program is thus in $O(|S|^3)$. □

A.4 Proof of Lemma 10

Proof. The proof is by a simple induction on the structure of ϕ. The interesting cases are when ϕ is $\exists x \psi(x, \overline{p})$ or $\forall x \psi(x, \overline{p})$. In the first case, if the translation holds then the value obtained by evaluating the witnessing function satisfies ψ, and so ϕ holds. Conversely, if ϕ holds then either 0 or 1 satisfies ψ, and it is easy to see that the witnessing function will evaluate to a satisfying value.

In the second case, observe that the witnessing function will falsify ψ if possible. Thus if the translation holds then ψ is satisfied by both 0 and 1. Conversely, if ϕ holds then ψ is satisfied no matter what the witnessing function evaluates to.

Note that the correctness of the cases of $\vee$ and $\wedge$ depends on the fact that whenever two Boolean programs are merged while translating a QBF, they never disagree on the definition of any function symbol. Thus, for example, the value of $\ulcorner\psi \vee \theta\urcorner$ with respect to $P[\psi \vee \theta]$ is indeed the Boolean $\vee$ of the values of $\ulcorner\psi\urcorner$ and $\ulcorner\theta\urcorner$, each with respect to their own Boolean programs. □

A.5 Proof of Lemma 11

First another lemma:

Lemma 23. *There are polynomial-size proofs of*

$$\overline{A} = \overline{B} \longrightarrow C(\overline{A}) = C(\overline{B}),$$

for any formulas $\overline{A}$,$\overline{B}$,C, in either language/system. In the case of BPLK, the size of P, the Boolean program defining the function symbols occurring in A, B, C, is an argument to the polynomial.

Proof. The proof is a simple induction on the structure of C.

- If C is a variable, then we need to prove $\overline{A} = \overline{B} \longrightarrow A_i = B_i$ for some i, which is trivial.
- If C is $D \wedge E$, $D \vee E$ or $\neg D$, we prove $\overline{A} = \overline{B} \longrightarrow D(\overline{A}) = D(\overline{B})$ and $\overline{A} = \overline{B} \longrightarrow E(\overline{A}) = E(\overline{B})$ and perform some simple propositional reasoning to obtain the desired sequent.
- If C is $QxD(x)$ for some quantifier Q, we first prove $\overline{A} = \overline{B} \longrightarrow D(\overline{A}, p) = D(\overline{B}, p)$. Some propositional reasoning yields $\overline{A} = \overline{B}, D(\overline{A}, p) \longrightarrow D(\overline{B}, p)$ and then two quantifier introductions give $\overline{A} = \overline{B}, QxD(\overline{A}, x) \longrightarrow QxD(\overline{B}, x)$. Likewise, we obtain $\overline{A} = \overline{B}, QxD(\overline{B}, x) \longrightarrow QxD(\overline{A}, x)$ and then some propositional reasoning yield the result.
- If C is $f_j(D_1, ..., D_k)$ then first prove

  $$\overline{A} = \overline{B} \longrightarrow \overline{D(\overline{A})} = \overline{D(\overline{B})}$$

  then apply **subst** to the sequent

  $$\overline{p} = \overline{q} \longrightarrow f_j(\overline{p}) = f_j(\overline{q})$$

  (which is proved by a simple induction on j) to substitute $\overline{D(\overline{A})}$ for p and $\overline{D(\overline{B})}$ for $\overline{q}$ and then **cut** to obtain the result.

□

Proof (Proof of lemma 11). We prove the lemma by induction on the structure of A. The base case is if A is atomic. If A is s, then $\ulcorner A\urcorner$ is s, so $\ulcorner A[B/s]\urcorner \equiv \ulcorner B\urcorner \equiv \ulcorner A\urcorner[\ulcorner B\urcorner/s]$. Otherwise A is some other variable u so $\ulcorner A[B/s]\urcorner \equiv \ulcorner A\urcorner[\ulcorner B\urcorner/s] \equiv u$. In either case, apply lemma 22.

If A is not atomic then we have the following cases for each possible main connective:

- A is of the form $C \vee D$: By assumption, we have proofs of $\ulcorner C[B/s]\urcorner \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s]$ and $\ulcorner D[B/s]\urcorner \longrightarrow \ulcorner D\urcorner[\ulcorner B\urcorner/s]$. Use **weakening** and $\vee$: **right** to produce new sequents with succedent $\ulcorner C\urcorner[\ulcorner B\urcorner/s] \vee \ulcorner D\urcorner[\ulcorner B\urcorner/s] \equiv \ulcorner(C \vee D)\urcorner[\ulcorner B\urcorner/s]$, and then apply $\vee$: **left**. The converse sequent is proved similarly.
- The case for $\wedge$ is symmetric and that for $\neg$ is similarly easy.
- A is $QxC(x, s, \overline{p}, \overline{r})$: In this case, we have already found a proof π_1 of $\ulcorner C[B/s]\urcorner \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s]$, and a proof π_2 of the converse sequent. We must find proofs of $\ulcorner A[B/s]\urcorner \longrightarrow \ulcorner A\urcorner[\ulcorner B\urcorner/s]$ and its converse. Now, the first step is to derive
$$\longrightarrow \ulcorner A[B/s]\urcorner = \ulcorner C[B/s]\urcorner[\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})/x] \tag{1}$$
and
$$\begin{aligned}\longrightarrow \ulcorner A\urcorner[\ulcorner B\urcorner/s] = \ulcorner C\urcorner[\epsilon_A(s, \overline{p}, \overline{r})/x][\ulcorner B\urcorner/s] \\ \equiv \ulcorner C\urcorner[\ulcorner B\urcorner/s][\epsilon_A(\ulcorner B\urcorner, \overline{p}, \overline{r})/x],\end{aligned} \tag{2}$$
which follow directly from the defining equations for $\ulcorner A[B/s]\urcorner \equiv f_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})$ and $\ulcorner A\urcorner \equiv f_A(s, \overline{p}, \overline{r})$, respectively.
The next step is to derive
$$\longrightarrow \epsilon_{A[B/s])}(\overline{p}, \overline{q}, \overline{r}) = \epsilon_A(\ulcorner B\urcorner, \overline{p}, \overline{r}). \tag{3}$$
First, use **subst** on the endsequent of π_1 from the induction hypothesis to substitute 1 (or 0, as appropriate for Q) for x to obtain the sequent $\ulcorner C[B/s]\urcorner[1/x](\overline{p}, \overline{q}, \overline{r}) \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s][1/x](\overline{p}, \overline{q}, \overline{r})$. Then, $\epsilon_{A[B/s]}$ introduction is applied on the left and ϵ_A introduction on the right. The process is repeated with π_2 and the introduction rules used on the opposite sides of the sequent, and the equality follows.
The final step is as follows: First substitute $\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})$ for x in the endsequent of π_1. We obtain a proof of
$$\ulcorner C[B/s]\urcorner[\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})/x] \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s][\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})/x]. \tag{4}$$
Then use the equality (3) derived in the previous paragraph with lemma 23 to obtain a proof of
$$\ulcorner C\urcorner[\ulcorner B\urcorner/s][\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})/x] \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s][\epsilon_A(\ulcorner B\urcorner, \overline{p}, \overline{r})/x]. \tag{5}$$
Now, using (4) and (5) we can easily obtain
$$\ulcorner C[B/s]\urcorner[\epsilon_{A[B/s]}(\overline{p}, \overline{q}, \overline{r})/x] \longrightarrow \ulcorner C\urcorner[\ulcorner B\urcorner/s][\epsilon_A(\ulcorner B\urcorner, \overline{p}, \overline{r})/x].$$
The desired sequent $\ulcorner A[B/s]\urcorner \longrightarrow \ulcorner A\urcorner[\ulcorner B\urcorner/s]$ is then obtained using (1) and (2).
The proof of the other desired sequent is obtained symmetrically to the final step above, but starting with π_2.

By induction therefore, we can obtain π_1' and π_2', proofs of the desired sequents, in polynomial time. $\square$

A.6 A Helper Lemma for Section 4

Lemma 24. *The sequent*

$$\ulcorner A \urcorner [\ulcorner B \urcorner / x] \longrightarrow \ulcorner A \urcorner [\epsilon_{\exists x A}(\overline{p})/x].$$

has a polynomial-size derivation.

Proof. This sequent has a proof involving 4 applications of lemma 23. In this proof sketch we omit the corner brackets and the subscript on $\epsilon_{\exists x A}$. We first show that either 1 satisfies A or not. Next, if 1 satisfies A then ϵ must, since in this case ϵ will evaluate to one. We then show that if 1 does not satisfy A, but B does, then ϵ must also, since in this case B and ϵ will both evaluate to 0. Finally, we combine these three pieces to get the result.

1.	$\epsilon(\overline{p}) := A[1/x]$	*BP Def'n*
2.	$\longrightarrow A[1/x], \neg A[1/x]$	*Easy*
3.	$\epsilon(\overline{p}) \longrightarrow A[1/x]$	*1*
4.	$A[1/x] \longrightarrow \epsilon(\overline{p})$	*1*
5.	$A[1/x], 1 \longrightarrow \epsilon(\overline{p})$	*4,* **wk.**
6.	$A[1/x], \epsilon(\overline{p}) \longrightarrow 1$	$\longrightarrow 1$, **wk.**
7.	$A[1/x], A[1/x] \longrightarrow A[\epsilon(\overline{p})/x]$	*5, 6, lem. 23*
8.	$A[1/x] \longrightarrow A[\epsilon(\overline{p})/x]$	*7,* **cntr.**
9.	$\neg A[1/x], \epsilon(\overline{p}) \longrightarrow 0$	*3,* $\neg$: **left**, **wk.**
10.	$\neg A[1/x], 0 \longrightarrow \epsilon(\overline{p})$	$0 \longrightarrow$, **wk.**
11.	$\neg A[1/x], A[0/x] \longrightarrow A[\epsilon(\overline{p})/x]$	*9, 10, lem. 23*
12.	$B, B \longrightarrow 1$	$\longrightarrow 1$, **wk.**
13.	$B, 1 \longrightarrow B$	$B \longrightarrow B$, **wk.**
14.	$B, A[B/x] \longrightarrow A[1/x]$	*12,13, lem. 23*
15.	$\neg A[1/x], A[B/x], B \longrightarrow 0$	*14,*$\neg$: **left**, **wk.**
16.	$\neg A[1/x], A[B/x], 0 \longrightarrow B$	$0 \longrightarrow$, **wk.**
17.	$\neg A[1/x] A[B/x], A[B/x] \longrightarrow A[0/x]$	*15, 16, lem. 23*
18.	$\neg A[1/x], A[B/x] \longrightarrow A[\epsilon(\overline{p})/x]$	*17,* **cntr.**, *11,* **wk.**, **cut**
19.	$A[B/x] \longrightarrow A[\epsilon(\overline{p})/x]$	*18, 8, 2,* **wk.**, **cut**

□

A.7 Example Translation of a Boolean Program into a QBF

Example 25. Consider the following simple contrived Boolean program:

$$f_1(p_1, p_2) := \neg p_1 \wedge p_2$$
$$f_2(p_1, p_2) := f_1(p_1, \neg p_2) \vee \neg f_1(f_1(\neg p_2, p_1), p_1)$$

Then by our terminology, $A_1(\overline{p})$ is $\neg p_1 \wedge p_2$, and $A_2(\overline{p})$ is $f_1(p_1, \neg p_2) \vee \neg f_1(f_1(\neg p_2, p_1), p_1)$. ϕ_0 is 1. Now, In the definition of f_1 there are no occurrences of function symbols, so there are no B_j. We thus define

$$\phi_1(z_{1,1}, z_{1,2}, u_1) := [(\neg z_{1,1} \wedge z_{1,2}) = u_1 \wedge [\phi_0 \supset [1]]].$$

ϕ_1 has a number of degenerate portions and is therefore not so interesting. In the definition of f_2, f_1 occurs 3 times (and is the only function symbol present). By our terminology we have the following:

$$
\begin{aligned}
m &= 3\\
j_1 &= 1\\
j_2 &= 1\\
j_3 &= 1\\
B_{1,1}(\overline{p}) &:= p_1\\
B_{1,2}(\overline{p}) &:= \neg p_2\\
B_{2,1}(\overline{p}) &:= f_1(\neg p_2, p_1)\\
B_{2,2}(\overline{p}) &:= p_1\\
B_{3,1}(\overline{p}) &:= \neg p_2\\
B_{3,2}(\overline{p}) &:= p_1\\
\widehat{B_{1,1}}(\overline{z_2}, \overline{v}) &:= z_{2,1}\\
\widehat{B_{1,2}}(\overline{z_2}, \overline{v}) &:= \neg z_{2,2}\\
\widehat{B_{2,1}}(\overline{z_2}, \overline{v}) &:= v_{f_1(\neg z_{2,2}, z_{2,1})}\\
\widehat{B_{2,2}}(\overline{z_2}, \overline{v}) &:= z_{2,1}\\
\widehat{B_{3,1}}(\overline{z_2}, \overline{v}) &:= \neg z_{2,2}\\
\widehat{B_{3,2}}(\overline{z_2}, \overline{v}) &:= z_{2,1}\\
\widehat{A_2}(\overline{z_2}, \overline{v}) &:= v_{f_1(z_{2,1}, \neg z_{2,2})} \vee \neg v_{f_1(f_1(\neg z_{2,2}, z_{2,1}), z_{2,1})}
\end{aligned}
$$

Now, ϕ_2 is constructed from the above as indicated in the definition:

$$
\begin{aligned}
&\phi_2(\overline{z_1}, \overline{z_2}, u_1, u_2) :=\\
&\quad \exists v_{f_1(z_{2,1}, \neg z_{2,2})}, v_{f_1(f_1(\neg z_{2,2}, z_{2,1}), z_{2,1})}[(v_{f_1(z_{2,1}, \neg z_{2,2})} \vee\\
&\quad\quad \neg v_{f_1(f_1(\neg z_{2,2}, z_{2,1}), z_{2,1})}) = u_2 \wedge\\
&\quad\quad \forall \overline{z'_1}, u'_1[\phi_1(\overline{z'_1}, u'_1) \supset [\\
&\quad\quad\quad (z'_{1,1} = z_{2,1} \wedge z'_{1,2} = \neg z_{2,2} \supset\\
&\quad\quad\quad\quad u'_1 = v_{f_1(z_{2,1}, \neg z_{2,2})}) \wedge\\
&\quad\quad\quad (z'_{1,1} = v_{f_1(\neg z_{2,2}, z_{2,1})} \wedge z'_{1,2} = z_{2,1} \supset\\
&\quad\quad\quad\quad u'_1 = v_{f_1(f_1(\neg z_{2,2}, z_{2,1}), z_{2,1})}) \wedge\\
&\quad\quad\quad (z'_{1,1} = \neg z_{2,2} \wedge z'_{1,2} = z_{2,1} \supset\\
&\quad\quad\quad\quad u'_1 = v_{f_1(\neg z_{2,2}, z_{2,1})}) \wedge\\
&\quad\quad\quad (z'_{1,1} = z_{1,1} \wedge z'_{1,2} = z_{1,2} \supset u'_1 = u_1)\\
&\quad\quad\quad]\\
&\quad\quad]\\
&\quad].
\end{aligned}
$$

A.8 Proof of Lemma 16

Proof. First, the statement vacuously holds for $i = 0$.

Now suppose it holds for $i - 1$. If $\phi_i(\overline{\overline{z}}, \overline{u})$ holds, then there exist v's satisfying the part of ϕ_i marked (*), which ensures that they have the same values as the function symbol occurrences they replace, so indeed $f_i(\overline{z_i}) = u_i$. The conjuncts (**) ensure that $f_j(\overline{z_j}) = u_j, j < i$.

Conversely, if $f_1(\overline{z_1}) = u_1 \wedge ... \wedge f_i(\overline{z_i}) = u_i$ holds, then the v's satisfying (*) (which exist and are unique) must have the correct values and so $\widehat{A_i}(\overline{z_i}, \overline{v}) = u_i$. Also, (**) is clearly satisfied, and thus all of ϕ_i is. □

A.9 Proof of Lemma 18

Proof. First note that for any BPLK formula ϕ, we have $|\widehat{\phi}|, |\ulcorner\phi\urcorner| \in O(|\phi|)$. These operators add a constant number of symbols for each replacement they perform, and this number is bounded by the size of the formula.

Next, consider the construction of ϕ_i from ϕ_{i-1}. The following are added:

- 2 copies of $\widehat{A_i}$
- 2 copies of $\widehat{B}$, for each B which is the argument to a function symbol in A_i (in the subsection (*))
- 3 occurrences of the corresponding v variables (in the subsection (*) and the quantifier)
- subsection (**) whose size is in $O(|P|)$.

Therefore summing these all up for ϕ_0 through ϕ_k we see that the last item dominates the sum and that $|\phi_k| \in O(|P|^2)$.

Finally, $\ulcorner S\urcorner$ consists of the prefix, at most $|\pi|$ occurrences of ϕ_k, each with substitutions of size at most $|\pi|$, followed by the suffix, of size $O(|S|)$. Therefore $|\ulcorner S\urcorner| \in O(|P|^2|\pi|^2)$. □

A.10 Proof of Lemma 19

Proof. These two lemmas are proved by induction in parallel.

For $i = 0$, the result is trivial.

Now assume the two lemmas are proved for $i - 1$. Let $\overline{\overline{B}} = \overline{B_1}, ..., \overline{B_m}$ be all formulas appearing as arguments to function symbols in the definition of f_i, $\overline{B_w}$ as arguments to f_{j_w}. Existence and uniqueness for ϕ_{i-1} plus some propositional reasoning give

$$
\begin{array}{l}
\longrightarrow \exists\overline{v}[\forall\overline{z'_1}, ..., \overline{z'_{i-1}}, u'_1, ..., u'_{i-1}[\phi_{i-1}(\overline{\overline{z'}}, \overline{u'}) \supset [\\
\qquad\qquad \vdots\\
\qquad\qquad (\overline{z'_{j_w}} = \overline{\widehat{B_w}} \supset u'_{j_w} = v_{f_{j_w}(\overline{B_w})})\wedge\\
\qquad\qquad \vdots\\
\qquad\qquad\qquad (\overline{z'_j} = \overline{z_j} \supset u'_j = u_j)\wedge\\
\qquad\qquad\qquad \vdots\\
\qquad\qquad\qquad]\\
\qquad\qquad]\\
\quad].
\end{array}
$$

Some more propositional reasoning (simply conjoining the tautology $\widehat{A_i}(\overline{z_i},\overline{v_i}) = \widehat{A_i}(\overline{z_i},\overline{v_i})$ inside the outermost quantifier) give

$$
\begin{array}{l}
\longrightarrow \exists \overline{v}[\widehat{A_i}(\overline{z_i},\overline{v_i}) = \widehat{A_i}(\overline{z_i},\overline{v_i}) \wedge \\
\qquad \forall \overline{z'_1},...,\overline{z'_{i-1}}, u'_1,...,u'_{i-1}[\phi_{i-1}(\overline{\overline{z'}},\overline{u'}) \supset [\\
\qquad\quad \vdots \\
\qquad\quad (\overline{z'_{j_w}} = \widehat{B_w} \supset u'_{j_w} = v_{f_{j_w}(\widehat{B_w})}) \wedge \\
\qquad\quad \vdots \\
\qquad\quad (\overline{z'_j} = \overline{z_j} \supset u'_j = u_j) \wedge \\
\qquad\quad \vdots \\
\qquad\quad] \\
\qquad] \\
],
\end{array}
$$

and then $\exists$: **right** (on the u's and one instance of $\widehat{A_i}$) and $\forall$: **right** (on the z's) yield the existence sequent for ϕ_i.

Now in the case of uniqueness, note that

$$\phi_i(\overline{\overline{z}},\overline{u}) \longrightarrow \phi_{i-1}(\overline{\overline{z}},\overline{u})$$

has a short proof using existence and uniqueness for $i-1$ and some propositional reasoning. Thus,

$$
\begin{array}{l}
\longrightarrow \phi_i(\overline{\overline{z_1}},\overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}},\overline{u_2}) \supset [\\
\quad (\overline{z_{1,1}} = \overline{z_{2,1}} \supset u_{1,1} = u_{2,1}) \wedge \\
\quad \vdots \\
\quad (\overline{z_{1,i-1}} = \overline{z_{2,i-1}} \supset u_{1,i-1} = u_{2,i-1}) \\
\quad]
\end{array}
\qquad (*)
$$

follows by uniqueness for $i-1$. Now we proceed as follows:

First by the definition of ϕ_i,

$$\phi_i(\overline{\overline{z_1}},\overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}},\overline{u_2}) \longrightarrow \exists \overline{v}[\widehat{A_i}(\overline{z_{1,i}},\overline{v}) = u_{1,i} \wedge ...] \wedge \exists \overline{v}[\widehat{A_i}(\overline{z_{2,i}},\overline{v}) = u_{2,i} \wedge ...].$$

Then renaming the quantified variables and doing some propositional reasoning,

$$\phi_i(\overline{\overline{z_1}},\overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}},\overline{u_2}) \longrightarrow \exists \overline{v_1},\overline{v_2}[[\widehat{A_i}(\overline{z_{1,i}},\overline{v_1}) = u_{1,i} \wedge ...] \wedge [\widehat{A_i}(\overline{z_{2,i}},\overline{v_2}) = u_{2,i} \wedge ...]].$$

Uniqueness for $i-1$ and more reasoning allows us to prove that the v's in one of the conjuncts are equal to those in the other, and thus produce

$$\phi_i(\overline{\overline{z_1}},\overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}},\overline{u_2}) \longrightarrow \exists \overline{v_1},\overline{v_2}[[\widehat{A_i}(\overline{z_{1,i}},\overline{v_1}) = u_{1,i} \wedge ...] \wedge [\widehat{A_i}(\overline{z_{2,i}},\overline{v_1}) = u_{2,i} \wedge ...]].$$

We can similarly consolidate the z's by adding a hypothesis:

$$\phi_i(\overline{\overline{z_1}},\overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}},\overline{u_2}) \longrightarrow \overline{z_{1,i}} = \overline{z_{2,i}} \supset \exists \overline{v_1},\overline{v_2}[\widehat{A_i}(\overline{z_{1,i}},\overline{v_1}) = u_{1,i} \wedge \widehat{A_i}(\overline{z_{1,i}},\overline{v_1}) = u_{2,i}].$$

Contracting,

$$\phi_i(\overline{\overline{z_1}}, \overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}}, \overline{u_2}) \longrightarrow \overline{z_{1,i}} = \overline{z_{2,i}} \supset \exists \overline{v_1}, \overline{v_2}[u_{1,i} = u_{2,i}].$$

We can now drop the quantifier:

$$\phi_i(\overline{\overline{z_1}}, \overline{u_1}) \wedge \phi_i(\overline{\overline{z_2}}, \overline{u_2}) \longrightarrow \overline{z_{1,i}} = \overline{z_{2,i}} \supset [u_{1,i} = u_{2,i}].$$

Some propositional reasoning to combine this last sequent with (*), and then $\neg$: **right** and several applications of $\forall$: **right** produce the uniqueness sequent for i. $\square$

A.11 Proof of Lemma 20

Proof. The first step is to use propositional reasoning to rename all the variables t in $\ulcorner T \urcorner(\ulcorner \psi \urcorner)$. A variable $t_{B(p)}$ is renamed to $t_{B(\psi)}$ by an application of lemma 5. This renaming can be done in any order, and call the resulting sequent U. Now, it is easy to see that for every occurrence of a subformula of the form $\ulcorner C(p) \urcorner$ in $\ulcorner T \urcorner$, the corresponding occurrence in U is $\ulcorner C(\psi) \urcorner$: This follows because whenever the translation operator replaces a subformula $B(p)$ of $C(p)$ by a function symbol, the symbol's name is $t_{B(p)}$, and so after the renaming it will be $t_{B(\psi)}$ as it should be.

Now, consider any variable $t_{f_i(\overline{B(p)})}$ occurring in $\ulcorner T \urcorner$. This variable is defined by an occurrence of ϕ_k in the prefix of $\ulcorner T \urcorner$:

$$\phi_k(0, ..., 0, \overline{\ulcorner B(p) \urcorner}, 0, ..., 0, d_1^1, ..., d_{i-1}^1, t_{f_i(\overline{B(p)})}, d_{i+1}^1, ..., d_k^1).$$

(In fact, it is possible that this variable occurs only in the prefix.) After the substitution of $\ulcorner \psi \urcorner$ into $\ulcorner T \urcorner$, the corresponding occurrence became

$$\phi_k(0, ..., 0, \overline{\ulcorner B(\overline{p}) \urcorner(\ulcorner \psi \urcorner)}, 0, ..., 0, d_1^1, ..., d_{i-1}^1, t_{f_i(\overline{B(p)})}, d_{i+1}^1, ..., d_k^1).$$

After the renaming, in U this occurrence becomes

$$\phi_k(0, ..., 0, \overline{\ulcorner B(\overline{\psi}) \urcorner}, 0, ..., 0, d_1^1, ..., d_{i-1}^1, t_{f_i(\overline{B(\psi)})}, d_{i+1}^1, ..., d_k^1),$$

which correctly defines $t_{f_i(\overline{B(\psi)})}$.

Now before the final step, note that the suffix of U is identical to the suffix of $\ulcorner T(\psi) \urcorner$, and those occurrences of ϕ_k defining t variables in the suffix of $\ulcorner T(\psi) \urcorner$ also occur in U. The only difference, then, between U and $\ulcorner T(\psi) \urcorner$ is that the former sequent may have some prefix formulas that the latter does not, and vice versa. We can thus use the existence sequents (or **contraction**, in the case of a duplicate) to cut away the superfluous prefix formulas from U, and **weakening** to add the missing ones. The result is the desired sequent. $\square$