Subsystems of BPLK

`$Id: subsys-bplk.tex,v 1.7 2003/12/17 06:01:50 alan Exp $` LaTeX'd on January 3, 2005

Alan Skelley

January 3, 2005

1 Introduction

In this note we discuss a way of defining subsystems of BPLK which are possibly equivalent in power to the subsystems G_i and G_i^* of G. We do this by presenting a hierarchy of Boolean function symbols which does correspond precisely in computational power to the levels of the Σ^q hierarchy of quantified propositional formulas.

Definition 1.1 (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 which defines functions. Whenever we refer to a BPLK-proof, we shall always explicitly write it as the pair* $< \pi, P >$ *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*

$$f(\overline{p}) := A(\overline{p}),$$

the new rules

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

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.*

The standard translations of G to and from BPLK from [2] are not precise with respect to quantifier alternations and in fact a G_1^* proof translated into BPLK and back could have arbitrarily large quantifier complexity. The problem seems to be that even a simple definition

$$f_2(\overline{x}) := f_1(f_1(\overline{x}))$$

or

$$f_5(\overline{x}) := (f_1(\overline{x}) \vee f_2(\overline{x})) \wedge (f_3(\overline{x}) \vee f_4(\overline{x}))$$

apparently requires several alternations of propositional quantifiers to translate properly without incurring exponential blowup when repeated.

2 Restrictions of Boolean Programs

A solution to the above problem is to define some sharply restricted subsystems of BPLK. First, we define a hierarchy of Boolean program function symbols:

Definition 2.1. Σ_i^{bp} *and* Π_i^{bp} *are defined as follows:*

1. *A function symbol f defined as*
$$f(\overline{x}) := \phi(\overline{x}),$$
with $\phi(\overline{x})$ purely propositional, is Σ_0^{bp} and Π_0^{bp}.

2. $(\Sigma_i^{bp} \bigcup \Pi_i^{bp}) \subseteq (\Sigma_{i+1}^{bp} \bigcap \Pi_{i+1}^{bp})$.

3. *If f is defined as*
$$f(\overline{x}) := g_1(\overline{x_1}) \vee ... \vee g_l(\overline{x_l})$$
with each $\overline{x_j}$ a list of variables and constants and each g_j is Σ_i^{bp}, then f is Σ_i^{bp}.

4. *If f is defined as*
$$f(\overline{x}) := g_1(\overline{x_1}) \wedge ... \wedge g_l(\overline{x_l})$$
where the $\overline{x_j}$ are as above and each g_j is Π_i^{bp}, then f is Π_i^{bp}.

Now, Σ_i^q formulas can be translated into equivalent Σ_i^{bp} function symbols by placing them in prenex form and translating quantifiers into disjunctions and conjunctions. Although the translation in [2] utilizes Hilbert ϵ-functions to translate quantifiers, we also noted an alternative translation suggested by Toniann Pitassi [1] which would translate quantifiers with conjunctions and disjunctions. Observe that these translations (restricted to prenex formulas) would actually produce function symbols in the desired level of the Σ^{bp} hierarchy.

The converse is also true, in the following sense: For a Boolean program defining a number of Σ_i^{bp} function symbols $f_1, ..., f_m$ there exists a single Σ_i^q formula $\psi(\overline{n}, \overline{c})$ with the property that $\psi(\ulcorner k \urcorner, \overline{c})$ holds exactly when $f_k(\overline{c}) = 1$, where $\ulcorner k \urcorner$ is the number k in binary, padded as required to match the number of variables in $\overline{n}$, which must obviously be enough to write $\ulcorner m \urcorner$. Before giving the final construction, observe the following points:

1. First, if $f_1, ..., f_l$ are Σ_0^{bp} function symbols defined as $f_j(\overline{x}) := \phi_j(\overline{x})$, then the desired formula is
$$\psi(\overline{n}, \overline{c}) := (\overline{n} = \ulcorner 1 \urcorner \wedge \phi_1(\overline{c})) \vee ... \vee (\overline{n} = \ulcorner l \urcorner \wedge \phi_l(\overline{c})),$$
which is clearly Σ_0^q.

2. Next, assume for the purposes of induction that ψ is a Σ_i^q translation of several Σ_i^{bp} function symbols in the above sense. Consider a new Σ_i^{bp} function symbol:
$$g_j(\overline{x}) := g_1(\overline{x_1}) \vee ... \vee g_l(\overline{x_l}),$$

where $g_1 ... g_l$ are all translated by ψ. Then we can define the Σ_i^q formula ψ' as follows to translate all the function symbols of ψ and additionally g_j:

$$\begin{aligned}&\psi'(\overline{n}, \overline{c}) := \exists\overline{n}'\exists\overline{c}'[\psi(\overline{n}', \overline{c}') \wedge [\\ &\quad [\overline{n} = \ulcorner j \urcorner \wedge [(\overline{n'} = \ulcorner 1 \urcorner \wedge \overline{c}' = \overline{x_1}) \vee ... \vee (\overline{n}' = \ulcorner l \urcorner \wedge \overline{c}' = \overline{x_l})]]] \vee\\ &\quad [\overline{n}' = \overline{n} \wedge \overline{c}' = \overline{c}]\\ &]].\end{aligned}$$

3. Similarly, suppose that
$$g_j(\overline{x}) := g_1(\overline{x_1}) \wedge ... \wedge g_l(\overline{x_l}),$$
is a new Π_i^{bp} where $g_1 ... g_l$ are all translated by the Π_i^q formula ψ. Then we can define the Π_i^q formula ψ' as follows to translate all the function symbols of ψ and additionally g_j:

$$\begin{aligned}&\psi'(\overline{n}, \overline{c}) := \forall\overline{n}'\forall\overline{c}'[[\\ &\quad [\overline{n}' = \overline{n} \wedge \overline{c}' = \overline{c}] \vee\\ &\quad [\overline{n} = \ulcorner j \urcorner \wedge [(\overline{n}' = \ulcorner 1 \urcorner \wedge \overline{c}' = \overline{x_1}) \vee ... \vee (\overline{n}' = \ulcorner l \urcorner \wedge \overline{c}' = \overline{x_l})]]\\ &] \supset \psi(\overline{n}', \overline{c}')].\end{aligned}$$

Now the construction is as follows: First, using point 1 above, translate all the Σ_0^{bp} function symbols with one Σ_0^q formula ψ_0. Next, translate all Σ_1^{bp} function symbols with one Σ_1^q formula ψ_1^+ by repeating point 2 above. Similarly, translate all Π_1^{bp} function symbols with one Π_1^q formula ψ_1^-. Finally, define
$$\psi_1(\overline{n}, \overline{c}) := \psi_1^+(\overline{n}, \overline{c}) \vee \psi_1^-(\overline{n}, \overline{c})$$
which is $\Sigma_2^q \bigcap \Pi_2^q$. (A simple disjunction is appropriate here since the sense of our translations is that the formula ψ, when supplied an invalid function symbol number, will be false). This process may now be repeated for the Σ_2^{bp} and Π_2^{bp} function symbols and each subsequent level of the hierarchy as required. The size of the translation is linear in the size of the original Boolean program if all symbols in the program are in a constant level $\Sigma_k^{bp} \bigcup \Pi_k^{bp}$ of the hierarchy. At each new level in the construction the previous translation is doubled in size (As it is used for both disjuncts, ψ_i^+ and ψ_i^-) and so more accurately the size of the translation is linear with a factor at least 2^k.

An alternative method to avoid even this factor is as follows: First form ψ_0, ψ_1^+ and ψ_1^- as above. As before, assume that ψ_i^+ (respectively, ψ_i^-) translates all Σ_i^{bp} (resp., Π_i^{bp}) function symbols. Form ψ_{i+1}^+ starting with ψ_i^- and repeating point 2 above to translate first the Σ_i^{bp} and next the Σ_{i+1}^{bp} function symbols. Likewise, form ψ_{i+1}^- starting from ψ_i^+ and applying point 3. In this way the doubling of size at each alternation is avoided. This latter translation may however complicate the translation of proofs in that there will be two different translations of Σ_i^{bp} function symbols: That provided by ψ_i^+ and also that provided by a subformula of ψ_{i+1}^+. It would probably be necessary to formalize the equivalence of these two translations in order to translate proofs.

3 Subsystems of BPLK

We are now able to define the subsystems of BPLK:

Definition 3.1. *$BPLK_i$ is the subsystem of BPLK in which every function symbol is $\Sigma_i^{bp} \bigcup \Pi_i^{bp}$ and additionally every formula containing rank-i function symbols is a valid defining formula for a $\Sigma_i^{bp} \bigcup \Pi_i^{bp}$ function symbol. Additionally, $BPLK_i^*$ is the subsystem of BPLK restricted to treelike proofs.*

Conjecture 3.2. *Although we do not prove it here, we believe that it should be straightforward to prove that BPLK$_i$ p-simulates G_i for the case when the endsequent is in prenex form.*

This would be done in two steps: First, the simulation from [2] would be revised using Pitassi's alternate translation. Then, a proof in G_i would be translated line-by-line by first converting every formula to prenex form in a canonical way and then applying the translation of formulas. As in the original translation from [2], there would be many technical details required to fill in the gaps in the translated proof, since the names of function symbols would change after the introduction of quantifiers. A further difficulty, not present in the original translation, is that now the introduction of a propositional connective may also cause the names of function symbols to change. Therefore, while the original translation required no extra steps to simulate a propositional inference, the current translation will require a derivation. This derivation will be similar in nature to a proof that a formula is equivalent to its prenex form.

Question 3.3. *Does BPLK$_i^*$ p-simulate G_i^*?*

The sub-derivations in the original translation are all treelike, and in fact most are proofs of sequents of the form $A \longrightarrow B$ when in fact A and B are equivalent. An obvious obstacle, though, is how to translate the introduction of a quantifier without using **subst** twice on the same sequent, and thus producing a non-treelike proof.

The simulations in the other direction seem to require many details and so for the moment we can only pose the question:

Question 3.4. *Does G_i p-simulate BPLK$_i$? What about their treelike subsystems?*

4 Conclusions

We originally posed the problem of stratifying BPLK in the hope that a solution might shed light on the structure of G and BPLK; it seems, however, that it might simply be a technical exercise without merit.

References

[1] Toniann Pitassi. Private communication, 2000.

[2] 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.