

arXiv:cs.PL/0408045 v1 19 Aug 2004

## On computing the fixpoint of a set of boolean equations

Viktor Kuncak

K. Rustan M. Leino

MIT

Microsoft Research

[vkuncak@mit.edu](mailto:vkuncak@mit.edu)

[leino@microsoft.com](mailto:leino@microsoft.com)

30 December 2003

Technical Report

MSR-TR-2003-08

This paper presents a method for computing a least fixpoint of a system of equations over booleans. The resulting computation can be significantly shorter than the result of iteratively evaluating the entire system until a fixpoint is reached.

Microsoft Research  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

<http://www.research.microsoft.com>

## 0 Introduction

Many problems in computer science, in particular those arising in the context of program analysis, involve the computation of a least (or, dually, greatest) fixpoint of a system of equations. In this paper, we consider a way to compute a least fixpoint when the equations involved are over the booleans. In some important cases, the resulting computation can be significantly shorter than the computation that iteratively evaluates the entire system until a fixpoint is reached.

Let us begin with an overview of our result. We restrict our attention to a finite lattice. A finite lattice is a complete lattice and has no infinite ascending chains, and any monotonic function on such a lattice is also continuous. Hence, the Kleene Fixpoint Theorem [2] states that the least fixpoint of any monotonic function  $F$  is the lattice join of the sequence of elements

$$F^0(\perp), F^1(\perp), F^2(\perp), \dots$$

where exponentiation denotes successive function applications and  $\perp$  denotes the bottom element of the lattice. Because this sequence is ascending and because the lattice is finite, there exists a natural number  $K$  such that

$$F^K(\perp)$$

is the least fixpoint of  $F$ . We call the least such  $K$  the *fixpoint depth* of  $F$ .

If we are able to evaluate function  $F$  and if we are able to determine whether two given lattice elements are equal, then we can compute the least fixpoint of  $F$ : starting from the value  $\perp$ , repeatedly apply  $F$  until the application of  $F$  leaves the value unchanged. The existence of a fixpoint depth guarantees that this process terminates. In this paper, we consider the problem of computing an *expression* for the least fixpoint, without computing the *value* of the expression. By first computing a small expression for the least fixpoint, we can relegate the computation of the value of the expression to an external tool such as a SAT solver [3]. In the sequel we therefore do not assume that we are able to compute the value of an expression into a particular lattice element.

The fixpoint depth of a function  $F$  on a lattice is bounded by the height of the lattice. Therefore, for the 2-element lattice  $\mathbb{B}$  of the booleans (which has height 1), the least fixpoint of  $F$  is given by  $F(\perp)$ , and for the  $2^n$ -element lattice  $\mathbb{B}^n$  that is the Cartesian product space of  $n$  booleans (which has height  $n$ ), the least fixpoint of  $F$  is given by  $F^n(\perp)$ .

Any function  $F: \mathbb{B}^n \rightarrow \mathbb{B}^n$  can be represented isomorphically by  $n$  functions  $f_i: \mathbb{B}^n \rightarrow \mathbb{B}$ . We write

$$F = (f_1, \dots, f_n)$$

where the tuple of functions is itself defined to be a function, as follows, for any  $n$ -tuple  $X$  of booleans:

$$(f_1, \dots, f_n)(X) = (f_1(X), \dots, f_n(X))$$

For example, let  $n = 3$  and let  $F = (f, g, h)$ . Then, the least fixpoint of  $F$  equals  $F^3(\perp, \perp, \perp)$ , as we have argued above. In terms of the functions  $f, g, h$ , this expands to:

$$\begin{aligned} & (f(f(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad g(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad h(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp))), \\ & g(f(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad g(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad h(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp))), \\ & h(f(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad g(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)), \\ & \quad h(f(\perp, \perp, \perp), g(\perp, \perp, \perp), h(\perp, \perp, \perp)))) \end{aligned}$$

We refer to this closed form of the fixpoint as the *Expanded Closed Form*. A different way to write down the Expanded Closed Form, which shares common subexpressions, is:

$$\begin{aligned} & \text{let } a_1 = \perp, \quad a_2 = \perp, \quad a_3 = \perp \quad \text{in} \\ & \text{let } b_1 = f(a_1, a_2, a_3), \quad b_2 = g(a_1, a_2, a_3), \quad b_3 = h(a_1, a_2, a_3) \quad \text{in} \\ & \text{let } c_1 = f(b_1, b_2, b_3), \quad c_2 = g(b_1, b_2, b_3), \quad c_3 = h(b_1, b_2, b_3) \quad \text{in} \\ & \text{let } d_1 = f(c_1, c_2, c_3), \quad d_2 = g(c_1, c_2, c_3), \quad d_3 = h(c_1, c_2, c_3) \quad \text{in} \\ & \quad (d_1, d_2, d_3) \end{aligned}$$

This representation is cubic in  $n$ , which means that computing it may take time and space that is cubic in  $n$ .\*

---

\*If we allow ourselves to write functions of  $n$  arguments as functions over  $n$ -tuples, then we can obtain a quadratic representation. For example, with  $n = 3$ , we have **let**  $a_1 = \dots, a_2 = \dots, a_3 = \dots$  **in** **let**  $a = (a_1, a_2, a_3)$  **in** **let**  $b_1 = f(a), b_2 = g(a), b_3 = h(a)$  **in** **let**  $b = (b_1, b_2, b_3)$  **in**  $\dots$

Let us consider another closed form, which we call the *Pruned Closed Form*. In the Pruned Closed Form, an application of a function  $f_i$  is replaced by  $\perp$  if it occurs in another application of the same function  $f_i$ . For the example above, where  $n = 3$ , the Pruned Closed Form is:

$$\begin{aligned}
 & (f(\perp, \\
 & \quad g(\perp, \perp, \quad h(\perp, \perp, \perp)), \\
 & \quad h(\perp, g(\perp, \perp, \perp), \perp \quad )), \\
 & g(f(\perp, \perp, \quad h(\perp, \perp, \perp)), \\
 & \quad \perp, \\
 & \quad h(\perp, \perp, \quad \perp \quad )), \\
 & h(f(\perp, g(\perp, \perp, \perp), \perp \quad )), \\
 & \quad g(\perp, \perp, \quad \perp \quad )), \\
 & \quad \perp \quad )),
 \end{aligned}$$

If we do not have any interpretation for the functions  $f_i$ —in other words, if each  $f_i$  is just a symbolic name for an uninterpreted function—then the cubic-sized Expanded Closed Form may be a reasonably small closed-form representation of the fixpoint. The Pruned Closed Form is generally much larger than cubic in  $n$ : for every subset  $S$  of  $f_2, \dots, f_n$ , function  $f_1$  appears expanded in a context where the set of enclosing functions is  $S$ . (A smaller Pruned Closed Form can be obtained by taking advantage of common subexpressions.) However, there are cases where the Pruned Closed Form can be significantly smaller than the Expanded Closed Form, for example when the fixpoint computation is dominated by the computation of local fixpoints, meaning fixpoints that involve only a small number of the functions. An important situation in program analysis where this case applies is when each function represents a control point in a given program, a function is defined in terms of the functions corresponding to the successor (or predecessor) control points, and the given program contains many local loops.

For example, suppose

$$f(x, y, z) = \mathfrak{f}(x, y) \quad g(x, y, z) = \mathfrak{g}(x) \quad h(x, y, z) = \mathfrak{h}(y, z)$$

for some functions  $\mathfrak{f}$ ,  $\mathfrak{g}$ , and  $\mathfrak{h}$ . Then the Expanded Closed Form is

$$\begin{aligned}
 & \text{let } a_1 = \perp, \quad a_2 = \perp, \quad a_3 = \perp \quad \text{in} \\
 & \text{let } b_1 = \mathfrak{f}(a_1, a_2), \quad b_2 = \mathfrak{g}(a_1), \quad b_3 = \mathfrak{h}(a_2, a_3) \quad \text{in} \\
 & \text{let } c_1 = \mathfrak{f}(b_1, b_2), \quad c_2 = \mathfrak{g}(b_1), \quad c_3 = \mathfrak{h}(b_2, b_3) \quad \text{in} \\
 & \text{let } d_1 = \mathfrak{f}(c_1, c_2), \quad d_2 = \mathfrak{g}(c_1), \quad d_3 = \mathfrak{h}(c_2, c_3) \quad \text{in} \\
 & \quad (d_1, d_2, d_3)
 \end{aligned}$$

In contrast, the Pruned Closed Form yields the much shorter expression

$$\begin{aligned} & (\mathbf{f}(\mathbf{f}(\perp, \mathbf{g}(\perp)), \\ & \quad \mathbf{g}(\perp)), \\ & \quad \mathbf{g}(\mathbf{f}(\perp, \perp)), \\ & \quad \mathbf{h}(\mathbf{g}(\mathbf{f}(\perp, \perp)), \\ & \quad \perp)) \end{aligned}$$

More generally, for an even  $n$ , suppose  $f_i(x_1, \dots, x_n)$  is  $f_i(x_i, x_{i+1})$  when  $i$  is odd and  $f_i(x_{i-1}, x_i)$  when  $i$  is even. Then the Expanded Closed Form is still cubic, whereas the Pruned Closed Form is the linear-sized expression

$$\begin{aligned} & ( \quad \mathbf{f}_1(\perp, \mathbf{f}_2(\perp, \perp)), \quad \mathbf{f}_2(\mathbf{f}_1(\perp, \perp), \perp), \\ & \quad \dots \\ & \quad \mathbf{f}_i(\perp, \mathbf{f}_{i+1}(\perp, \perp)), \quad \mathbf{f}_{i+1}(\mathbf{f}_i(\perp, \perp), \perp), \\ & \quad \dots \\ & \quad \mathbf{f}_{n-1}(\perp, \mathbf{f}_n(\perp, \perp)), \quad \mathbf{f}_n(\mathbf{f}_{n-1}(\perp, \perp), \perp) \\ & ) \end{aligned}$$

In the rest of this paper, we define the Pruned Closed Form more precisely and prove that it yields the same value as the Expanded Closed Form.

## 1 Using the Bekić-Leszczylowski Theorem

In this section, we sketch how to obtain the Pruned Closed Form by applications of the Bekić-Leszczylowski Theorem [1, 4].

We write

$$(\downarrow x \bullet R(x))$$

for the lattice meet of all values for  $x$  that satisfy the predicate  $R(x)$ . For any monotonic function  $F$ , we then write

$$(\downarrow x \bullet x = F(x)) \tag{0}$$

to denote the least fixpoint of  $F$ , because the Tarski Fixpoint Theorem [5] says that the meet of all fixpoints is itself a fixpoint. Using for a function  $F: \mathbb{B}^n \rightarrow \mathbb{B}^n$  the isomorphic representation of  $n$  functions  $f_i: \mathbb{B}^n \rightarrow \mathbb{B}$ , we can write (0) equivalently as:

$$\begin{aligned} (\downarrow x_1, \dots, x_n \bullet \quad & x_1 = f_1(x_1, \dots, x_n) \quad \wedge \\ & \vdots \quad \wedge \\ & x_n = f_n(x_1, \dots, x_n) ) \end{aligned}$$

We can now state the Bekić-Leszczylowski Theorem [1, 4], for any monotonic functions  $F$  and  $G$  (possibly over different lattices):

$$\begin{aligned} & (\downarrow a, b \bullet a = F(a, b) \wedge b = G(a, b) ) \\ = & \\ & (\downarrow a, b \bullet a = F(a, b) \wedge b = (\downarrow b \bullet b = G(a, b) )) \end{aligned}$$

Note that each side of the equality expresses a fixpoint in the lattice  $\mathbb{B}^n$  if  $F$  and  $G$  are functions of types  $\mathbb{B}^p \times \mathbb{B}^q \rightarrow \mathbb{B}^p$  and  $\mathbb{B}^p \times \mathbb{B}^q \rightarrow \mathbb{B}^q$ , respectively, for  $p$  and  $q$  such that  $p + q = n$ .

A consequence of the Bekić-Leszczylowski Theorem and the Kleene Fixpoint Theorem for a known fixpoint depth is the following lemma:

**Lemma 0** For any lattice domain  $\mathbb{A}$  and monotonic functions  $F: \mathbb{A} \times \mathbb{B} \rightarrow \mathbb{A}$  and  $G: \mathbb{A} \times \mathbb{B} \rightarrow \mathbb{B}$ ,

$$\begin{aligned} & (\downarrow a, b \bullet a = F(a, b) \wedge b = G(a, b) ) \\ = & \\ & (\downarrow a, b \bullet a = F(a, b) \wedge b = G(a, \perp) ) \end{aligned}$$

*Proof.*

$$\begin{aligned} & (\downarrow a, b \bullet a = F(a, b) \wedge b = G(a, b) ) \\ = & \quad \{ \text{Bekić-Leszczylowski Theorem} \} \\ & (\downarrow a, b \bullet a = F(a, b) \wedge b = (\downarrow b \bullet b = G(a, b) )) \\ = & \quad \{ (\lambda b \bullet G(a, b)) \text{ is a function on } \mathbb{B}, \text{ and} \\ & \quad \text{therefore its fixpoint depth is at most 1, and} \\ & \quad \text{therefore } (\downarrow b \bullet b = G(a, b)) = G(a, \perp) \} \\ & (\downarrow a, b \bullet a = F(a, b) \wedge b = G(a, \perp) ) \quad \blacksquare \end{aligned}$$

Using Lemma 0, we now show that the Pruned Closed Form is indeed the least fixpoint in  $\mathbb{B}^2$ . For any monotonic boolean functions  $f$  and  $g$ :

$$\begin{aligned} & (\downarrow a, b \bullet a = f(a, b) \wedge b = g(a, b) ) \tag{1} \\ = & \quad \{ \text{Lemma 0 with } F, G := f, g \} \\ & (\downarrow a, b \bullet a = f(a, b) \wedge b = g(a, \perp) ) \\ = & \quad \{ \text{substitute equals for equals} \} \\ & (\downarrow a, b \bullet a = f(a, g(a, \perp)) \wedge b = g(a, \perp) ) \\ = & \quad \{ \text{Lemma 0 with} \\ & \quad G, F := (\lambda a, b \bullet f(a, g(a, \perp))), (\lambda a, b \bullet g(a, \perp)) \} \\ & (\downarrow a, b \bullet a = f(\perp, g(\perp, \perp)) \wedge b = g(a, \perp) ) \end{aligned}$$

This calculation shows that an expression for the least solution of  $a$  in equation (1) is

$$f(\perp, g(\perp, \perp))$$

By a symmetric argument, an expression for the least solution of  $b$  in equation (1) is

$$g(f(\perp, \perp), \perp)$$

That is, an expression for (1) is

$$( f(\perp, g(\perp, \perp)), g(f(\perp, \perp), \perp) )$$

which is the Pruned Closed Form.

Using the result for  $\mathbb{B}^2$ , we can show that the Pruned Closed Form is also the least fixpoint in  $\mathbb{B}^3$ . For any monotonic boolean functions  $f$ ,  $g$ , and  $h$ :

$$\begin{aligned}
& (\downarrow a, b, c \bullet a = f(a, b, c) \wedge b = g(a, b, c) \wedge c = h(a, b, c)) & (2) \\
= & \{ \text{Lemma 0 with } G := h \text{ (and with } F \text{ as the isomorphic} \\
& \quad \text{representation of functions } f \text{ and } g) \} \\
& (\downarrow a, b, c \bullet a = f(a, b, c) \wedge b = g(a, b, c) \wedge c = h(a, b, \perp)) \\
= & \{ \text{substitute equals for equals} \} \\
& (\downarrow a, b, c \bullet a = f(a, b, c) \wedge b = g(a, b, h(a, b, \perp)) \wedge c = h(a, b, \perp)) \\
= & \{ \text{Lemma 0 with } G := (\lambda a, b, c \bullet g(a, b, h(a, b, \perp))) \} \\
& (\downarrow a, b, c \bullet a = f(a, b, c) \wedge b = g(a, \perp, h(a, \perp, \perp)) \wedge c = h(a, b, \perp)) \\
= & \{ \text{substitute equals for equals} \} \\
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), c) \wedge \\
& \quad b = g(a, \perp, h(a, \perp, \perp)) \wedge c = h(a, b, \perp)) \\
= & \{ \text{the first 3 steps of this calculation, in reverse order} \} \\
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), c) \wedge \\
& \quad b = g(a, b, c) \wedge c = h(a, b, c)) \\
= & \{ \text{Lemma 0 with } G := g \} \\
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), c) \wedge \\
& \quad b = g(a, \perp, c) \wedge c = h(a, b, c)) \\
= & \{ \text{substitute equals for equals} \} \\
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), c) \wedge \\
& \quad b = g(a, \perp, c) \wedge c = h(a, g(a, \perp, c), c)) \\
= & \{ \text{Lemma 0 with } G := (\lambda a, b, c \bullet h(a, g(a, \perp, c), c)) \}
\end{aligned}$$

$$\begin{aligned}
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), c) \wedge \\
& \quad b = g(a, \perp, c) \wedge c = h(a, g(a, \perp, \perp), \perp) ) \\
= & \quad \{ \text{substitute equals for equals} \} \\
& (\downarrow a, b, c \bullet a = f(a, g(a, \perp, h(a, \perp, \perp)), h(a, g(a, \perp, \perp), \perp)) \wedge \\
& \quad b = g(a, \perp, c) \wedge c = h(a, g(a, \perp, \perp), \perp) ) \\
= & \quad \{ \text{Lemma 0 with } G := \\
& \quad (\lambda a, b, c \bullet f(a, g(a, \perp, h(a, \perp, \perp)), h(a, g(a, \perp, \perp), \perp)) ) \} \\
& (\downarrow a, b, c \bullet a = f(\perp, g(\perp, \perp, h(\perp, \perp, \perp)), h(\perp, g(\perp, \perp, \perp), \perp)) \wedge \\
& \quad b = g(a, \perp, c) \wedge c = h(a, g(a, \perp, \perp), \perp) )
\end{aligned}$$

This calculation shows that an expression for the least solution of  $a$  in (2) is

$$f(\perp, g(\perp, \perp, h(\perp, \perp, \perp)), h(\perp, g(\perp, \perp, \perp), \perp))$$

and similarly for  $b$  and  $c$ .

Our main result is that the Pruned Closed Form is the least fixpoint in  $\mathbb{B}^n$  for any  $n$ . In the next section, we prove this result directly, not using Lemma 0.

## 2 The theorem

We are given  $n \geq 1$  monotonic functions  $f_1, \dots, f_n: \mathbb{B}^n \rightarrow \mathbb{B}$ , where  $\mathbb{B}$  is the boolean domain  $\{0, 1\}$  ordered by  $\leq$  (with  $0 \leq 1$ ). To represent an indexed  $n$ -tuple of things, like a list of booleans  $x_1, \dots, x_n$ , we write  $\vec{x}$ . The fact that the given functions are monotonic is written as follows, for any index  $i$  and any tuples of booleans  $\vec{x}$  and  $\vec{y}$ :

$$\vec{x} \vec{\leq} \vec{y} \Rightarrow f_i \cdot \vec{x} \leq f_i \cdot \vec{y}$$

where an infix dot (with the highest operator precedence) denotes function application, and the order  $\vec{\leq}$  is the component-wise ordering of tuples:

$$\vec{x} \vec{\leq} \vec{y} \equiv (\forall i \bullet x_i \leq y_i)$$

We are interested in viewing the functions as specifying a system of equations, namely:

$$\begin{aligned}
x_1, \dots, x_n : \quad x_1 &= f_1 \cdot (x_1, \dots, x_n) \\
&\dots \\
x_n &= f_n \cdot (x_1, \dots, x_n)
\end{aligned} \tag{3}$$



where the variables to the left of the colon show the unknowns. We take a tuple of functions  $(f_1, \dots, f_n)$ , which we can also write as  $\vec{f}$ , to itself be a function, one which produces a tuple from the results of applying the given argument to each of the functions. For example, for the functions given above and an argument  $\vec{x}$ , we have:

$$(f_1, \dots, f_n) \cdot \vec{x} = (f_1 \cdot \vec{x}, \dots, f_n \cdot \vec{x})$$

Thus, we can write the system (3) of equations as:

$$\vec{x} : \vec{x} = \vec{f} \cdot \vec{x}$$

We are interested in the *least* (in the sense of the ordering  $\succeq$ ) solution  $\vec{x}$  that satisfies this equation. That is, we are interested in the *least fixpoint* of the function  $\vec{f}$ . Because the lattice of boolean  $n$ -tuples has height  $n$ , the least fixpoint of  $\vec{f}$  can be reached by applying  $\vec{f}$   $n$  times starting from the bottom element of the lattice. That is, the least fixpoint of  $\vec{f}$  is given by:

$$\vec{f}^n \cdot \vec{0}$$

where exponentiation denotes successive function applications and  $\vec{0}$  is the tuple of  $n$  0's.

To precisely specify the Pruned Closed Form, we introduce a notation that keeps track of which functions have been applied in the enclosing context. In particular, we use a set that contains the indices of the functions already applied. Formally, we define the following family of functions, for any index  $i$  and set  $S$  of indices:

$$g_{S,i} = \begin{cases} f_i \circ (g_{S \cup \{i\},1}, \dots, g_{S \cup \{i\},n}) & \text{if } i \notin S \\ (\lambda \vec{x} \bullet 0) & \text{if } i \in S \end{cases}$$

Taking advantage of our previous notation and using  $\dot{0}$  to denote the function that always returns 0 (that is, the boolean 0 extended pointwise to a boolean function), we can write the definition of  $g$  as follows:

$$g_{S,i} = \begin{cases} f_i \circ \overline{g_{S \cup \{i\}}} & \text{if } i \notin S \\ \dot{0} & \text{if } i \in S \end{cases}$$

Our goal is now to prove the following:

**Theorem 1**  $\quad \overline{g_{\emptyset}} \cdot \vec{0} = \vec{f}^n \cdot \vec{0}$

### 3 Proof

We start by proving some lemmas that we use in the proof of this theorem.

**Lemma 2** For any index  $i$  and for any  $S \subsetneq \{1, \dots, n\}$ ,

$$g_{S,i} \cdot \vec{0} \leq (f_i \circ \vec{f}^{n-|S|-1}) \cdot \vec{0}$$

*Proof.* By induction on  $n - |S|$ . Let  $T$  denote  $S \cup \{i\}$ . We consider three cases.

CASE  $i \in S$ :

$$\begin{aligned} & g_{S,i} \cdot \vec{0} \\ = & \{ \text{definition of } g, \text{ since } i \in S \} \\ & \dot{0} \cdot \vec{0} \\ = & \{ \text{definition of } \dot{0} \} \\ & 0 \\ \leq & \{ 0 \text{ is bottom element of } \leq \} \\ & (f_i \circ \vec{f}^{n-|S|-1}) \cdot \vec{0} \end{aligned}$$

CASE  $i \notin S \wedge |T| < n$ :

$$\begin{aligned} & g_{S,i} \cdot \vec{0} \\ = & \{ \text{definition of } g, \text{ since } i \notin S \} \\ & (f_i \circ \vec{g}_T) \cdot \vec{0} \\ = & \\ & (f_i \circ (g_{T,1}, \dots, g_{T,n})) \cdot \vec{0} \\ = & \{ \text{distribute } \cdot \vec{0} \} \\ & f_i \cdot (g_{T,1} \cdot \vec{0}, \dots, g_{T,n} \cdot \vec{0}) \\ \leq & \{ \text{for each index } j, \text{ induction hypothesis with } i, S := j, T, \text{ since} \\ & |S| + 1 = |T| < n; \text{ and monotonicity of } f_i \} \\ & f_i \cdot ((f_1 \circ \vec{f}^{n-|S|-2}) \cdot \vec{0}, \dots, (f_n \circ \vec{f}^{n-|S|-2}) \cdot \vec{0}) \\ = & \{ \text{distribute } \cdot \vec{0} \} \\ & (f_i \circ (f_1 \circ \vec{f}^{n-|S|-2}, \dots, f_n \circ \vec{f}^{n-|S|-2})) \cdot \vec{0} \\ = & \{ \text{distribute } \circ \vec{f}^{n-|S|-2} \} \\ & (f_i \circ (f_1, \dots, f_n) \circ \vec{f}^{n-|S|-2}) \cdot \vec{0} \\ = & \{ \text{exponentiation} \} \\ & (f_i \circ \vec{f}^{n-|S|-1}) \cdot \vec{0} \end{aligned}$$

CASE  $i \notin S \wedge |T| = n$ :

$$\begin{aligned}
 & g_{S,i} \cdot \vec{0} \\
 = & \{ \text{see first 3 steps of previous case} \} \\
 & f_i \cdot (g_{T,1} \cdot \vec{0}, \dots, g_{T,n} \cdot \vec{0}) \\
 = & \{ \text{for each index } j, j \in T, \text{ so } g_{T,j} = \dot{0} \} \\
 & f_i \cdot \vec{0} \\
 = & \{ |S| = n - 1, \text{ so } f^{n-|S|-1} \text{ is the identity function} \} \\
 & (f_i \circ f^{n-|S|-1}) \cdot \vec{0} \quad \blacksquare
 \end{aligned}$$

The following corollary of Lemma 2 proves one direction of Theorem 1.

**Corollary 3**  $\vec{g}_\emptyset \cdot \vec{0} \preceq \vec{f}^n \cdot \vec{0}$

*Proof.*

$$\begin{aligned}
 & \vec{g}_\emptyset \cdot \vec{0} \\
 = & \\
 & (g_{\emptyset,1} \cdot \vec{0}, \dots, g_{\emptyset,n} \cdot \vec{0}) \\
 \preceq & \{ \text{for each index } j, \text{ Lemma 2 with } i, S := j, \emptyset \} \\
 & ((f_1 \circ \vec{f}^{n-1}) \cdot \vec{0}, \dots, (f_n \circ \vec{f}^{n-1}) \cdot \vec{0}) \\
 = & \{ \text{distribute } \cdot \vec{0} \text{ and } \circ \vec{f}^{n-1} \} \\
 & ((f_1, \dots, f_n) \circ \vec{f}^{n-1}) \cdot \vec{0} \\
 = & \{ \text{exponentiation} \} \\
 & \vec{f}^n \cdot \vec{0} \quad \blacksquare
 \end{aligned}$$

To support the remaining lemmas, we define one more family of functions. For any index  $i$  and set  $S$  of indices,

$$h_{S,i} = \begin{cases} f_i & \text{if } i \notin S \\ \dot{0} & \text{if } i \in S \end{cases}$$

**Lemma 4** For any index  $i$ , monotonic function  $H: \mathbb{B}^n \rightarrow \mathbb{B}^n$ , and  $m \geq 0$ ,

$$(f_i \circ H^m) \cdot \vec{0} = 0 \Rightarrow (\forall p \mid 0 \leq p \leq m \bullet (f_i \circ H^p) \cdot \vec{0} = 0)$$

*Proof.* We prove the term of the quantification as follows:

$$\begin{aligned}
& (f_i \circ H^p). \vec{0} \\
\leq & \quad \{ \text{monotonicity of } f_i \text{ and } H, \text{ since } \vec{0} \preceq H^{m-p}. \vec{0} \} \\
& (f_i \circ H^p).(H^{m-p}. \vec{0}) \\
= & \\
& (f_i \circ H^m). \vec{0} \\
= & \quad \{ \text{antecedent} \} \\
& 0 \quad \blacksquare
\end{aligned}$$

**Lemma 5** For any index  $i$ , set  $S$  of indices,  $m \geq 0$ , and  $T = S \cup \{i\}$ ,

$$(f_i \circ \vec{h}_S^m). \vec{0} = 0 \Rightarrow (\forall p \mid 0 \leq p \leq m \bullet \vec{h}_S^p. \vec{0} = \vec{h}_T^p. \vec{0})$$

*Proof.* If  $i \in S$ , then  $S = T$  and the consequent follows trivially. For  $i \notin S$ , we prove the term of the quantification by induction on  $p$ .

CASE  $p = 0$ : Trivial—exponentiation with 0 gives identity function.

CASE  $p > 0$ :

$$\begin{aligned}
& \vec{h}_S^p. \vec{0} \\
= & \quad \{ \text{exponentiation, since } p > 0 \} \\
& \vec{h}_S. (\vec{h}_S^{p-1}. \vec{0}) \\
= & \quad \{ \text{distribute } .(\vec{h}_S^{p-1}. \vec{0}) \} \\
& (h_{S,1}.(\vec{h}_S^{p-1}. \vec{0}), \dots, h_{S,n}.(\vec{h}_S^{p-1}. \vec{0})) \\
= & \quad \{ \text{for any index } j, h_{S,j}.(\vec{h}_S^{p-1}. \vec{0}) = h_{T,j}.(\vec{h}_S^{p-1}. \vec{0}), \text{ see below} \} \\
& (h_{T,1}.(\vec{h}_S^{p-1}. \vec{0}), \dots, h_{T,n}.(\vec{h}_S^{p-1}. \vec{0})) \\
= & \quad \{ \text{distribute } .(\vec{h}_S^{p-1}. \vec{0}) \} \\
& \vec{h}_T. (\vec{h}_S^{p-1}. \vec{0}) \\
= & \quad \{ \text{induction hypothesis with } p := p - 1 \} \\
& \vec{h}_T. (\vec{h}_T^{p-1}. \vec{0}) \\
= & \quad \{ \text{exponentiation} \} \\
& \vec{h}_T^p. \vec{0}
\end{aligned}$$

Now for the proof of the third step in the calculation above. If  $j \neq i$ , then  $j \in S \equiv j \in T$ , so  $h_{S,j} = h_{T,j}$ . If  $j = i$ , then:

$$\begin{aligned}
& h_{S,i}.(\vec{h}_S^{p-1}. \vec{0}) \\
= & \quad \{ \text{definition of } h, \text{ since } i \notin S \}
\end{aligned}$$

$$\begin{aligned}
 & f_i.(\vec{h}_S^{p-1} . \vec{0}) \\
 = & \quad \{ \text{Lemma 4 with } H, p := \vec{h}_S, p - 1, \text{ using the antecedent of} \\
 & \quad \text{Lemma 5 to fulfill the antecedent of Lemma 4} \} \\
 & 0 \\
 = & \quad \{ \text{definition of } \dot{0} \} \\
 & \dot{0}.(\vec{h}_S^{p-1} . \vec{0}) \\
 = & \quad \{ \text{definition of } h, \text{ since } i \in T \} \\
 & h_{T,i}.(\vec{h}_S^{p-1} . \vec{0}) \quad \blacksquare
 \end{aligned}$$

We need one more lemma.

**Lemma 6** For any index  $i$ , set  $S$  of indices, and  $m$  satisfying  $0 \leq m \leq n - |S|$ ,

$$(h_{S,i} \circ \vec{h}_S^m). \vec{0} \leq g_{S,i}. \vec{0} \quad (4)$$

*Proof.* By induction on  $m$ . We consider three cases.

CASE  $i \in S$ :

$$\begin{aligned}
 & h_{S,i} \circ \vec{h}_S^m \\
 = & \quad \{ \text{definition of } h, \text{ since } i \in S \} \\
 & \dot{0} \circ \vec{h}_S^m \\
 = & \quad \{ \dot{0} \text{ is left zero element of } \circ \} \\
 & \dot{0} \\
 = & \quad \{ \text{definition of } g, \text{ since } i \in S \} \\
 & g_{S,i}
 \end{aligned}$$

CASE  $i \notin S \wedge m = 0$ :

$$\begin{aligned}
 & (h_{S,i} \circ \vec{h}_S^m). \vec{0} \\
 = & \quad \{ \text{exponentiation, since } m = 0 \} \\
 & h_{S,i}. \vec{0} \\
 = & \quad \{ \text{definition of } h, \text{ since } i \notin S \} \\
 & f_i. \vec{0} \\
 \leq & \quad \{ \text{monotonicity of } f_i, \text{ since } \vec{0} \preceq \vec{g}_T. \vec{0} \} \\
 & f_i.(\vec{g}_T. \vec{0}) \\
 = & \quad \{ \text{definition of } g, \text{ since } i \notin S \} \\
 & g_{S,i}. \vec{0}
 \end{aligned}$$

CASE  $i \notin S \wedge m > 0$ : It suffices to prove that the left-hand side of (4) is 0 whenever the right-hand side is 0. Therefore, we assume the latter to be 0:

$$g_{S,i} \cdot \vec{0} = 0 \quad (5)$$

and prove the former to be 0:

$$\begin{aligned}
 & (h_{S,i} \circ \vec{h}_S^m) \cdot \vec{0} \\
 = & \quad \{ \text{definition of } h, \text{ since } i \notin S \} \\
 & (f_i \circ \vec{h}_S^m) \cdot \vec{0} \\
 = & \quad \{ \text{exponentiation, since } m > 0 \} \\
 & (f_i \circ (h_{S,1}, \dots, h_{S,n}) \circ \vec{h}_S^{m-1}) \cdot \vec{0} \\
 = & \quad \{ \text{distribute } \circ \vec{h}_S^{m-1} \text{ and } \cdot \vec{0} \} \\
 & f_i((h_{S,1} \circ \vec{h}_S^{m-1}) \cdot \vec{0}, \dots, (h_{S,n} \circ \vec{h}_S^{m-1}) \cdot \vec{0}) \\
 \leq & \quad \{ (6), \text{ see below; and monotonicity of } f_i \} \\
 & f_i(g_{T,1} \cdot \vec{0}, \dots, g_{T,n} \cdot \vec{0}) \\
 = & \\
 & (f_i \circ \vec{g}_T) \cdot \vec{0} \\
 = & \quad \{ \text{definition of } g, \text{ since } i \notin S \} \\
 & g_{S,i} \cdot \vec{0} \\
 = & \quad \{ \text{assumption (5)} \} \\
 & 0
 \end{aligned}$$

In this calculation, we used the following fact: for every index  $j$ ,

$$(h_{S,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} \leq g_{T,j} \cdot \vec{0} \quad (6)$$

which we now prove. We divide the proof of (6) up into two sub-cases.

SUB-CASE  $(h_{S,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} = 0$ : Formula (6) follows immediately.

SUB-CASE  $(h_{S,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} \neq 0$ : First, we derive some consequences of assumption (5):

$$\begin{aligned}
 & g_{S,i} \cdot \vec{0} = 0 \\
 \Rightarrow & \quad \{ \text{induction hypothesis with } S, i, m := S, i, m - 1 \} \\
 & (h_{S,i} \circ \vec{h}_S^{m-1}) \cdot \vec{0} = 0 \quad (7)
 \end{aligned}$$

$$\begin{aligned}
 = & \quad \{ \text{definition of } h, \text{ since } i \notin S \} \\
 & (f_i \circ \vec{h}_S^{m-1}) \cdot \vec{0} = 0 \\
 \Rightarrow & \quad \{ \text{Lemma 5 with } m, p := m - 1, m - 1 \} \\
 & \vec{h}_S^{m-1} \cdot \vec{0} = \vec{h}_T^{m-1} \cdot \vec{0} \quad (8)
 \end{aligned}$$

Now, calculating from the assumption we made in this sub-case:

$$\begin{aligned}
& (h_{S,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} \neq 0 \\
= & \quad \{ (7) \} \\
& (h_{S,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} \neq 0 \wedge i \neq j \\
\Rightarrow & \quad \{ i \neq j, \text{ so } j \in S \equiv j \in T, \text{ so } h_{S,j} = h_{T,j} \} \\
& (h_{T,j} \circ \vec{h}_S^{m-1}) \cdot \vec{0} \neq 0 \\
= & \quad \{ (8) \} \\
& (h_{T,j} \circ \vec{h}_T^{m-1}) \cdot \vec{0} \neq 0 \\
\Rightarrow & \quad \{ \text{induction hypothesis with } S, i, m := T, j, m - 1 \} \\
& g_{T,j} \cdot \vec{0} \neq 0 \\
\Rightarrow & \\
& (6)
\end{aligned}$$

This concludes the proof of Lemma 6.  $\blacksquare$

And finally, the proof of the theorem:

*Proof of Theorem 1.* The proof is a ping-pong argument.

$$\begin{aligned}
& \vec{g}_\emptyset \cdot \vec{0} \\
\lesssim & \quad \{ \text{Corollary 3} \} \\
& \vec{f}^n \cdot \vec{0} \qquad \text{---ping!} \\
= & \quad \{ \text{exponentiation, since } n \geq 1 \} \\
& ((f_1, \dots, f_n) \circ \vec{f}^{n-1}) \cdot \vec{0} \\
= & \quad \{ \text{distribute } \circ \vec{f}^{n-1} \text{ and } \cdot \vec{0} \} \\
& ((f_1 \circ \vec{f}^{n-1}) \cdot \vec{0}, \dots, (f_n \circ \vec{f}^{n-1}) \cdot \vec{0}) \\
= & \quad \{ \text{by definition of } h, h_{\emptyset,i} = f_i \text{ for each index } i; \text{ and} \\
& \quad \text{thus also } \vec{h}_\emptyset = \vec{f} \} \\
& ((h_{\emptyset,1} \circ \vec{h}_\emptyset^{n-1}) \cdot \vec{0}, \dots, (h_{\emptyset,n} \circ \vec{h}_\emptyset^{n-1}) \cdot \vec{0}) \\
\lesssim & \quad \{ \text{Lemma 6 with } S, i, m := \emptyset, i, n - 1 \text{ for each } i \} \\
& (g_{\emptyset,1} \cdot \vec{0}, \dots, g_{\emptyset,n} \cdot \vec{0}) \\
= & \quad \{ \text{distribute } \cdot \vec{0} \} \\
& \vec{g}_\emptyset \cdot \vec{0} \qquad \blacksquare \qquad \text{---pong!}
\end{aligned}$$

## 4 Related Work and Acknowledgments

Our theorem has already found a use, namely in the translation of boolean programs into satisfiability formulas [3].

Before we knew of the Bekić-Leszczylowski Theorem, one of us (Kuncak) proved the theorem as detailed in Section 3. Tony Hoare then proposed a way to prove the theorem in a way that would eliminate recursive uses of variables, one by one. In doing this, Hoare also proved what essentially amounts to the Bekić-Leszczylowski Theorem, appealing only to the Tarski Fixpoint Theorem [5]. We elaborated this format in Section 1, to whose formulation Carroll Morgan also contributed. We learnt about the Bekić-Leszczylowski Theorem from Patrick Cousot. The theorem is often called simply the Bekić Theorem, but de Bakker [0] traces an independent proof thereof to Leszczylowski. Finally, we are grateful for feedback from the Eindhoven Tuesday Afternoon Club and the participants of the IFIP WG 2.3 meeting in Biarritz, France (March 2003).

## References

- [0] J. de Bakker. *Mathematical Theory of Program Correctness*. Prentice-Hall, 1980.
- [1] Hans Bekić. Definable operation in general algebras, and the theory of automata and flowcharts. In Cliff B. Jones, editor, *Programming Languages and Their Definition—Hans Bekić (1936–1982)*, volume 177 of *Lecture Notes in Computer Science*, pages 30–55. Springer, 1984.
- [2] Stephen Cole Kleene. *Introduction to Metamathematics*. D. Van Nostrand, New York, 1952.
- [3] K. Rustan M. Leino. A SAT characterization of boolean-program correctness. In Thomas Ball and Sriram K. Rajamani, editors, *Model Checking Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 104–120. Springer, May 2003.
- [4] Jacek Leszczylowski. A theorem on resolving equations in the space of languages. *Bull. Acad. Polon. Sci.*, 19:967–970, 1971.
- [5] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.