Electronic Colloquium on Computational Complexity, Report No. 12 (2000)

Integer Circuit Evaluation is PSPACE-complete

Ke Yang *

October 22, 1999

Abstract

In this paper, we address the problem of evaluating the Integer Circuit (IC), or the $\{\cup, \times, +\}$-circuit over the set of natural numbers. The problem is a natural extension to the integer expression by Stockmeyer and Mayer, and is also studied by Mckenzie, Vollmer and Wagner in their "Polynomial Replacement System". We show a polynomial-time algorithm that reduces QBF (Quantified Boolean Formula) problem to the Integer Circuit problem. This complements the result of [W84] to show that IC problem is PSPACE-complete. The proof in this paper provides a new perspective to describe PSPACE-completeness.

1 Introduction

1.1 Background and Motivation

In many instances of complexity theory, "formulae vs. circuits" plays a crucial role. For example, polynomial-size boolean formula evaluation is in NC_1, while polynomial-size boolean circuit evaluation is P-complete. In as early as 1973, Stockmeyer and Meyer studied the complexity of integer expressions [SM73]. They showed that $\{\cup, +\}$-expression problem is NP-complete. Their result easily extends to the $\{\cup, \times, +\}$-expressions. Interestingly, if we allow negation, namely, consider the circuit $\{\cup, \times, +, \neg\}$, the problem is already PSPACE-complete. But they didn't consider the integer circuits.

The general problem on the complexity of Circuit evaluation has always been of people's interests. For instance, Beaudry-McK-Peladeau-Therien have a somewhat deep treatment of circuit evaluation over finite monoroids [BMPT97]. Allender et al. [AJMV98] discussed depth reduction.

Kaltofen [K88] discussed the difficulty with coefficient growth when polynomials are represented as straight-line programs (another view of arithmetic circuits). See also Plaisted's hardness result [P77].

Wagner in his paper [W84] defined "hierarchical description", which are the same as $\{\cup, +\}$-circuits. He proved (Lemma 1.2) that the membership problem for hierarchical descriptions is in PSPACE, and left open if this problem is SPACE-complete.

Mckenzie, Vollmer and Wagner in [MVW99] defined "polynomial replacement systems", and their paper naturally leads the question on the PSPACE-completeness of the integer circuit evaluation.

1.2 Result

In this paper, we prove that integer circuit, or $\{\cup, \times, +\}$-circuit evaluation is PSPACE-hard. This result, combined with [W84], shows that the integer circuit evaluation problem is PSPACE-complete. As a consequence, the EVAL[·] and RANGE[·] problems considered in [MVW99] for simple acyclic prs are PSPACE-complete.

There are PSPACE-completeness characterizations, like Stockmeyer and Meyer's QBF [SM73], Babai and Fortnow's Arithmetic Program with Binary Substitution [BF91], Shamir's IP = PSPACE[S90], Cai and Furst's Three-bit Bottleneck [CF87]. Our result provides a new perspective to describe PSPACE-completeness.

*Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsbugh, PA 15213, USA. E-mail: yangke@cmu.edu. The electronic version of the paper is available at http://gs89.sp.cs.cmu.edu/research/papers/pspace.ps

ISSN 1433-8092

1.3 Organization of the paper

Section 2 contains the definitions and formal notations that would be used in the rest of the paper. Section 3 contains the reduction from QBF to the Integer Circuit Problem. The reduction consists of 3 parts: pre-processing the QBF; core reduction to remove the quantifiers; and the post-processing to extract the result. Section 4 is the summary.

2 Definitions and notations

We give some definitions and notations are will be used in the rest of the paper.

2.1 Quantified Boolean Formula

QBF (Quantified Boolean Formula) is a well-known problem. We consider the following **standard form** of QBF:

$$Q_1 x_1 Q_2 x_2 ... Q_n x_n \ \phi(x_1, x_2, ..., x_n)$$

where $Q_i = \forall$ or $\exists$, $i = 1, 2, ..., n$, and $\phi(x_1, ..., x_n)$ is a DNF of variables $x_1, x_2, ..., x_n$. We call this type of QBF "standard", since it has special properties that:

1. all variables are quantified, namely, there is no "free" variables — we call such a QBF **closed**.
2. ϕ is a DNF, i.e., ϕ is the OR of several terms, while each term is an AND of literals.

The QBF problem is: given a QBF in standard form, to decide if the formula is true or false. It is well-understood that the standard QBF is a PSPACE-complete problem [SM73], [P94].

2.2 Integer Circuit

We consider the following problem: a *Integer Circuit* (IC) is a polynomial-sized circuit on finite sets of positive integers. It takes n variables: $X_1, X_2, ..., X_n$, where each X_n is a singleton, i.e., $X_i = \{x_i\}, x_i \in N$, for $i = 1, 2, ..., n$. The gates in the circuits are: **Union**, denoted by $\cup$, which takes the union of two sets; **Addition**, denoted by $\oplus$, defined as $A \oplus B := \{a + b \mid a \in A, b \in B\}$; **Multiplication**, denoted by $\otimes$, defined as $A \otimes B := \{a \cdot b \mid a \in A, b \in B\}$. So the circuit takes in n singletons, and outputs a finite set of positive integers. An *Integer Circuit Problem* (ICP) is, given such a circuit with the inputs, and a positive integer N, to decide if the final output set contains this N. Below is the formal definition of IC.

Definition 1 (IC & ICP) *An* **Integer Circuit** *(IC) family is a family of circuits $\{C_n\}$. For every n, C_n has n input gates and a number of computational gates of fan-in 2, one of which is denoted the "final gate". The total number of gates is bounded by a polynomial in n. Each input gate takes in one singleton of positive integer. All the computational gates are of fan-in 2 and perform operations on finite set of positive integers. There are three kind of internal gates:*

1. **Union Gate** *It is denoted by $\cup$, and computes the union of two input sets.*
2. **Addition Gate** *It is denoted by $\oplus$, and is defined as $A \oplus B := \{a + b \mid a \in A, b \in B\}$*
3. **Multiplication Gate** *It is denoted by $\otimes$, and is defined as $A \otimes B := \{a \cdot b \mid a \in A, b \in B\}$*

The output of the circuit is the output of the "final gate".
A **Integer Circuit Problem** *(ICP) is that given the triple $\langle C, x, N \rangle$, where C is an IC, x is the input to C, N is an integer, and we use $C(x)$ to denote the output of C on input x, to decide if $N \in C(x)$.*

Wagner have shown that ICP is in PSPACE [W84]. In this paper, we show it is actually PSPACE-complete. We prove this by showing a reduction from QBF to the ICP.

2.3 Integer Vector

We give some definitions on integer vectors and the operations on them.

Definition 2 (Integer Vector) *An* n**-dimensional integer vector** $\mathbf{v}$ *is written as* $\mathbf{v} = \langle v^1, v^2, ..., v^n \rangle$, *where* $v^i \in Z$, *for* $i = 1, 2, ...n$. *We denote the set of all* n*-dimensional integer vectors by* Z^n.

Definition 3 (L_∞ norm) *For integer vector* $\mathbf{v}$, *we define its* L_∞ **norm** *to be:*

$$\|\mathbf{v}\| := \max \{|v^i| \ : i = 1, 2, ..., n\}$$

For a finite set S *of integer vectors, we define its* L_∞ **norm** *to be:*

$$\|S\| := \max \{\|\mathbf{v}\| \ : \ \mathbf{v} \in S\}$$

Definition 4 (Neighbors) *Two* n*-dimensional integer vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are* **neighbors in the** k**-th entry**, *iff:*

1. $u^k = -v^k$.
2. $u^i = v^i$, *for all* $i \neq k, 1 \leq i \leq n$.

We denote this by $\mathbf{u} = [\mathbf{v}]_k$ *and* $\mathbf{v} = [\mathbf{u}]_k$.

Definition 5 (Scalar Multiple) *For integer* a *and integer vector* $\mathbf{v}$, *their* **scalar multiple** *is defined as*

$$a \cdot \mathbf{v} := \langle a \cdot v^1, a \cdot v^2, ..., a \cdot v^n \rangle$$

We denote $-1 \cdot \mathbf{v}$ *by* $-\mathbf{v}$.

Definition 6 (Addition and Multiplication of Integer Vectors) *For integer vectors* $\mathbf{u}$ *and* $\mathbf{v}$, *we define the addition and multiplication as follows:*

$$\begin{aligned} \mathbf{u} + \mathbf{v} &:= \langle u^1 + v^1, u^2 + v^2, ..., u^n + v^n \rangle \\ \mathbf{u} \cdot \mathbf{v} &:= \langle u^1 \cdot v^1, u^2 \cdot v^2, ..., u^n \cdot v^n \rangle \end{aligned}$$

Notice the multiplication defined here is entry-wise multiplication, rather than the inner product.
Now we define some useful constant vectors.

Definition 7 *We define the following constants:*

- $\mathbf{0}^n := \langle 0, 0, .., 0 \rangle$.
- $\mathbf{1}^n := \langle 1, 1, .., 1 \rangle$.
- $\mathbf{e}_k^n := \langle e_1, e_2, .., e_n \rangle$, *where* $e_k = 1$, *and* $e_i = 0$ *for all* $i \neq k, 1 \leq i \leq n$.
- $\mathbf{1}_k^n := \langle x_1, x_2, .., x_n \rangle$, *where* $x_k = -1$, *and* $x_i = 1$ *for all* $i \neq k, 1 \leq i \leq n$.

For example, $\mathbf{e}_2^5 = \langle 0, 1, 0, 0, 0 \rangle$, and $\mathbf{1}_3^4 = \langle 1, 1, -1, 1 \rangle$.
When the value of n is clear from the context and there is no danger of ambiguity, we normally eliminate the n — e.g., we write e_k rather than e_k^n.

2.4 Vector Integer Circuit

We will look at another kind of circuit — *Vector Arithmetic Circuit* (VIC), which is closely related to the IC. Actually the only difference between VIC and IC is that in VIC, the operations are over sets of *integer vectors*, rather than sets of *positive integers* in IC.

Definition 8 (VIC & VICP) *An* **Vector Integer Circuit** *(VIC) family is a family of circuits* $\{C_n\}$. *For every* n, C_n *has* n *input gates and a number of computational gates of fan-in 2, one of which is denoted the "final gate". The total number of gates is bounded by a polynomial in* n. *Each input gate takes in one singleton of integer vector. All the computational gates are of fan-in 2 and perform operations on finite set of integer vectors. There are three kind of internal gates:*

1. **Union Gate** *It is denoted by* $\cup$, *and computes the union of two input sets.*
2. **Addition Gate** *It is denoted by* $\oplus$, *and is defined as* $A \oplus B := \{\mathbf{u} + \mathbf{v} \mid u \in A, v \in B\}$

3. **Multiplication Gate** *It is denoted by* $\otimes$*, and is defined as* $A \otimes B := \{\mathbf{u} \cdot \mathbf{v} \mid u \in A, v \in B\}$

The output if the circuit is the output of the "final gate".

A **Vector Integer Circuit Problem** *(VICP) is that given the triple* $\langle C, x, \mathbf{v}\rangle$*, where* C *is a VIC,* x *is the input to* C*,* $\mathbf{v}$ *is an integer vector, and we use* $C(x)$ *to denote the output of* C *on input* x*, to decide if* $\mathbf{v} \in C(x)$*.*

So an vector integer circuit has the almost same structure as an integer circuit, but the elements under operation are integer vectors, rather than positive integers, and they have their version of addition and multiplication.

The reason we introduce the VICP is that we will first reduce the QBF to VICP, and then we further reduce VICP to ICP, thus showing ICP is PSPACE-hard.

3 Reducing QBF to ICP

In this section we show how to reduce QBF to VICP. Our reduction contains three parts: pre-pocessing the DNF to a truth table; core reduction to remove the quantifiers; and post-pocessing to extract the result. The first two parts reduce QBF to VICP, and the third part reduces VICP to ICP.

3.1 Part 1: Pre-processing the DNF

We begin by some definitions.

It is intuitively clear we can use integer vectors to represent truth assignments, and all we need is a formal definition.

Definition 9 (Truth assignment vector) *An* n*-dimensional integer vector* $\mathbf{v}$ *is a* **truth assignment vector***, if all its entries are 1 or -1, i.e.,* $v^i = \pm 1$*, for* $i = 1, 2, ...n$*.*

In the rest of the paper, when n is fixed and clear from the context, we normally refrain from writing n explicitly.

Claim 1 *There are* 2^n *truth assignment vectors in* Z^n*.*

Definition 10 (Satisfying vector) *For an* n*-dimensional truth assignment vector* $\mathbf{v}$ *and a boolean formula* ϕ *of* n *variables* $x_1, ..., x_n$*, we say* $\mathbf{v}$ **satisfies** ϕ*, if* $\phi(y_1, y_2, .., y_n)$ *is TRUE, where* $y_i = TRUE$ *if* $v^i = 1$*, and* $y_i = FALSE$ *if* $v^i = -1$*, for* $i = 1, 2, ..., n$*.*

Notice we can add "dummy variables" into a boolean formula, and thus a formula of n variables is also a formula of m variables for $m > n$. For example, we can view formula $(x_1 \wedge x_2) \vee \neg x_3$ as a formula of variables x_1, x_2, x_3, x_4, and so vector $\langle 1, 1, 1, -1\rangle$ is a satisfying vector for this formula. In particular, we can also view the constants TRUE and FALSE as boolean formulae of n variables. Therefore any n-dimensional truth assignment vector satisfies TRUE and no truth assignment vector satisfies FALSE.

Definition 11 (Truth Table) *For a formula* ϕ *of* n *variables, we define its* **truth table** *to be*

$$T(\phi) := \{\mathbf{v} \mid \mathbf{v} \texttt{ satisfies } \phi,\ \mathbf{v} \in Z^n\}$$

In particular, when ϕ *is a constant (TRUE or FALSE). Is truth table is the set of all* n*-dimensional truth assigment vectors when it is TRUE and* $\emptyset$ *when it is FALSE.*

Our goal in this part is to construct a truth table for a DNF, using the vector integer circuit.

Notice if ϕ is a DNF, then it can be written as

$$\phi = \alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_m$$

where each α_i is a conjunctive form. So we know

$$T(\phi) = \bigcup_{i=1}^{m} T(\alpha_i) \tag{1}$$

While each $T(\alpha_i)$ is easy to compute: WLOG we assume

$$\alpha_i = x_1 \wedge x_2 \wedge \cdots \wedge x_k \wedge \neg x_{k+1} \wedge \cdots \wedge \neg x_{k+l}$$

then

$$T(\alpha_i) = \{\langle \underbrace{1, 1, .., 1}_{k}, \underbrace{-1, -1, ..., -1}_{l}, 0, ..., 0\rangle\} \oplus \{\mathbf{e}_{k+l+1}, -\mathbf{e}_{k+l+1}\} \oplus \cdots \oplus \{\mathbf{e}_n, -\mathbf{e}_n\} \qquad (2)$$

here the $\oplus$ is the addition operation in vector integer circuit.
Thus we have

Lemma 1 *There is a polynomial time algorithm A_1, that takes in a DNF ϕ and outputs a vector integer circuit C along with its input x, such that $C(x) = T(\phi)$.*

Proof: The algorithm A_1 does the following: First it uses Equation 2 to compute $T(\alpha_i)$ for $i = 1, 2, .., m$, and then it uses Equation 1 to compute $T(\phi)$. The total size is polynomial in n and m, and thus is polynomial in $|\phi|$. ■

3.2 Part 2: Core reduction — remove the quantifiers

This part is the main part of the reduction.
Notice the QBF we are studying takes the form

$$F = Q_1 x_1 Q_2 x_2 ... Q_n x_n \ \phi(x_1, ..., x_n)$$

By Lemma 1, we know we can have a truth table for ϕ. Now we want to compute the truth table of F. Notice since F is **closed**, i.e. there is no free variables in F, its value is either TRUE or FALSE, and thus its truth table is either the set of all n-dimensional truth assignment vectors or the empty set.
From now on we will fix n, that is, for all boolean formulae, we will assume that they are formulae of n variables.
Let's look at the QBF with a single quantifier:

Definition 12 (Equivalent Formula) *Let $F = Q_m x_m \ \phi(x_1, ..., x_m)$ be a QBF with one quantifier, where ϕ is a boolean formula. We define its* **equivalent formula** *to be*

$$\phi' := \begin{cases} \phi(x_1, ..., x_{m-1}, 0) \wedge \phi(x_1, ..., x_{m-1}, 1) & \texttt{if } Q = \forall \\ \phi(x_1, ..., x_{m-1}, 0) \vee \phi(x_1, ..., x_{m-1}, 1) & \texttt{if } Q = \exists \end{cases}$$

which is also a boolean formula.[1]

Then we obviously have

Claim 2 *F and ϕ' are equivalent, i.e., for all $x_1, x_2, ..., x_{m-1} \in \{0, 1\}$, $F(x_1, ..., x_{m-1}) = \phi'(x_1, ..., x_{m-1})$.*

Lemma 2 *Let $F = Q_m x_m \ \phi(x_1, ..., x_m)$ be a QBF with one quantifier, and ϕ' be its equivalent Formula. We denote the truth table of ϕ by $T = T(\phi)$. Then we have*

- *If $Q_m = \forall$, then $T(\phi') = \{\mathbf{v} \ : \ \mathbf{v} \in T \wedge \ [\mathbf{v}]_m \in T\}$.*
- *If $Q_m = \exists$, then $T(\phi') = \{\mathbf{v} \ : \ \mathbf{v} \in T \vee \ [\mathbf{v}]_m \in T\}$.*

Proof: Immediate from the definition of ϕ and ϕ'.

■

Now, given a QBF with n quantifiers, we can remove the quantifiers one by one, from inside to outside.

Definition 13 (Equivalent QBF Chain) *Let $F = Q_1 x_1 Q_2 x_2 ... Q_n x_n \ \phi(x_1, ..., x_n)$ be a closed QBF. An* **equivalent QBF chain** *of F is a sequence of QBF's: $\{F_0, F_1, ..., F_n\}$, satisfying:*

[1] In the rest of the paper, when there is no danger of confusion, We will use 1 and 0 to represent TRUE and FALSE, respectively.

1. $F_n = F$.
2. $F_{k-1} = Q_1 x_1 ... Q_{k-1} x_{k-1}\ \phi_{k-1}$, *and* ϕ_{k-1} *is the equivalent Formula of* $Q_k x_k \phi_k$, *for* $k = 1, 2, ..., n$, *and* ϕ_n *is defined to be* ϕ.

So all the QBF's in the equivalent QBF chain are closed, and in particular, $F_0 = \phi_0$ is a constant of value either TRUE or FALSE. Again it is immediate that all the QBF's in the chain are equivalent. So F is true, iff F_0 is true, and iff the truth table of F_0 is the set of all n-dimensional truth assignment vectors.

Next we will see how we can compute the truth tables of ϕ_n, ϕ_{n-1}, ..., ϕ_0 using vector integer circuits, inductively. When we have the truth table of ϕ_0 at hand (it is either the complete set or the empty set), we can tell if the original formula F is TRUE or FALSE.

We define a transition function as follows:

Definition 14 (Transition Function) *The* **transition function** f *is defined as follows;*

$$f(Q, S, k, n) := \begin{cases} S \oplus (S \otimes \{\mathbf{1}_k^n\}) & \texttt{if } Q = \forall \\ 2 \cdot S \cup (2 \cdot S \otimes \{\mathbf{1}_k^n\}) & \texttt{if } Q = \exists \end{cases} \tag{3}$$

where Q *is a quantifier, and* S *is a set of integer vectors.* $\mathbf{1}_k^n$ *is defined as before. When there is no danger of confusion, we may omit the* n.

The transition function can be implemented by the vector integer circuit. Suppose now we have a QBF $F = Q_1 x_1 Q_2 x_2 ... Q_n x_n\ \phi(x_1, ..., x_n)$, where ϕ is a DNF, and $S = T(\phi)$ is the truth table of ϕ. Then we can repeatedly apply the transition f on S:

Definition 15 (Operation Chain) *For a closed QBF* F, *let* S *be the truth table of the DNF in* F. *We define* $S_n := S$, *and* $S_{k-1} := f(Q_k, S_k, k, n)$, *for* $k = n, n-1, ..., 1$. *The sequence* $\{S_0, S_1, ..., S_n\}$ *is called the* **operation chain** *for* F.

We have several observations:

Lemma 3 $\|S_k\| \leq 2^{n-k}$

This can be easily shown by induction on k.

Definition 16 (Good vectors) *An integer vector* $\mathbf{v} = \langle v^1, v^2, ..., v^n \rangle \in S_k$ *is* **good in** S_k, *iff* $|v^i| = 2^{n-k}$ *for all* $i = 1, 2, ..., n$.

The next lemma is an important one.

Lemma 4 *If* $\mathbf{v} \in S_{k-1}$ *is good, then:*

1. *If* $Q_k = \forall$, *then there exists* $\mathbf{u} \in S_k$, *such that:*
   - $\mathbf{u}$ *is good in* S_k.
   - $[\mathbf{u}]_k$, *which is also good, is also in* S_k.
   - $\mathbf{v} = 2 \cdot \mathbf{u}$.
   - $[\mathbf{v}]_k$, *which is also good, is also in* S_{k-1}.
2. *If* $Q_k = \exists$, *then there exists* $\mathbf{u} \in S_k$, *such that:*
   - $\mathbf{u}$ *is good in* S_k.
   - $\mathbf{v} = 2 \cdot \mathbf{u}$ *or* $\mathbf{v} = 2 \cdot \mathbf{u} \cdot \mathbf{1}_k$.
   - $[\mathbf{v}]_k$, *which is also good, is also in* S_{k-1}.

Proof: We consider two cases:

1. Suppose $Q_k = \forall$. Then, by Definition 14, there must exist vectors $\mathbf{u}, \mathbf{w} \in S_k$, such that $\mathbf{v} = \mathbf{u} + \mathbf{w} \cdot \mathbf{1}_k$. Notice $\mathbf{v}$ is good, so $|v^i| = 2^{n-k+1}$, for all $i = 1, 2, ..., n$. However, $v^i = u^i + w^i$ for $i \neq k$, and $v^k = u^k - w^k$. By Lemma 3, each entry of $\mathbf{u}$ and $\mathbf{w}$ is bounded by 2^{n-k}. Thus the only possibility is that both $\mathbf{u}$ and $\mathbf{w}$ are good in S_k, $u^i = w^i$, for $i \neq k$, and $u^k = -w^k$. That means $\mathbf{w} = [\mathbf{u}]_k$. So $\mathbf{v} = \mathbf{u} + \mathbf{w} \cdot \mathbf{1}_k = \mathbf{u} + [\mathbf{u}]_k \cdot \mathbf{1}_k = 2 \cdot \mathbf{u}$. Furthermore, $[\mathbf{v}]_k = \mathbf{w} + \mathbf{u} \cdot \mathbf{1}_k \in S_{k-1}$.

2. Suppose $Q_k = \exists$. Then, still by Definition 14, there exists a vector $\mathbf{u} \in S_k$, such that $\mathbf{v} = 2 \cdot \mathbf{u}$ or $\mathbf{v} = 2 \cdot \mathbf{u} \cdot \mathbf{1}_k$. In either case, this $\mathbf{u}$ should be good, and notice $[\mathbf{v}]_k = \mathbf{v} \cdot \mathbf{1}_k$. So we have $[\mathbf{v}]_k = 2 \cdot [\mathbf{u}]_k \cdot \mathbf{1}_k \in S_{k-1}$ or $[\mathbf{v}]_k = 2 \cdot \mathbf{u} \cdot \mathbf{1}_k \cdot \mathbf{1}_k = 2 \cdot \mathbf{u} \in S_{k-1}$.

■

What this lemma tells us is that: if there is a good vector $\mathbf{v}$ in S_{k-1}, then there must be good vector in S_k corresponding to it, and the neighbor of $\mathbf{v}$ in the k-th place is also in S_k.

In the subsequent discussion, we will only focus on the good vectors in $S_0, S_1, ..., S_n$.

Definition 17 (Filtering Function & Filtered Operation Chain) *We define, for every k, a* **filtering function**, *$\psi_k(\cdot)$, as follows:*

$$\psi_k(S_k) = \{\mathbf{v} \mid \mathbf{v} \in S_k, \ \mathtt{v\ is\ good\ in}\ S_k\}$$

For the operation chain $\{S_0, S_1, ..., S_n\}$ of a QBF F we define $G_k := \psi_k(S_k)$, for $k = 0, 1, ..., n$, and call the sequence $\{G_0, G_1, ..., G_n\}$ the **filtered operation chain** *of F.*

So $\psi_k(\cdot)$ "filters" all the non-good vectors out of S_k.

Then immediately from 4, we have

Claim 3 $G_k = \psi_k(f(Q_{k+1}, G_{k+1}, k))$.

The intuition is: there are two ways to do the filtering: one way is we filter out those non-good vectors in each step of the transition, and only apply the next transition to the remaining good ones; the other way if that we do all the transitions first, and then filter out the non-good ones in the end. What lemma 4 tells us is that these two ways have the same result.

Now we are ready to prove the main lemma, which connects our filtered operation chain $\{G_k\}$, to the truth tables of the equivalent QBF chain.

Lemma 5 (Main Lemma) *Let F be a QBF, and its equivalent QBF chain be $\{F_0, F_1, ..., F_n\}$. For each F_i, write its Formula inside the quantifiers by ϕ_i, and denote the truth table of ϕ_i by $T_i = T(\phi_i)$. Let $\{G_k\}$ be the filtered operation chain of F as defined in Definition 17. Then we have*

$$G_k = 2^{n-k} \cdot T_k$$

and especially, $G_0 = 2^n \cdot T_0$. Since T_0 is the truth table of ϕ_0, and thus T_0 is either the complete set of truth assignment vectors or the empty set, depending on whether F is TRUE or FALSE. Therefore G_0 is also either the complete set of the empty set.

So we have

Corollary 1 (All-or-Nothing Rule) *If the QBF F is TRUE, then G_0 contains all good vectors for S_0, namely, all 2^n vectors whose entries are $\pm 2^n$; if F is FALSE, then G_0 is empty.*

Proof: [Main Lemma]

We prove by induction.

For $k = n$, Notice both T_n and S_n are truth tables of ϕ, so $T_n = S_n$. Each vector in the truth table is a truth assignment vector, i.e., their entries are all ± 1. Therefore all vectors in S_n are good, and thus $G_n = S_n = T_n$.

If the lemma is true for k, now we look at the case $k-1$. We inspect the relationship between T_{k-1} and T_k, the truth table for the Formula ϕ_{k-1} and ϕ_k. ϕ_{k-1} is the equivalent Formula for $Q_k x_k \ \phi_k$. We look at the cases $Q_k = \forall$ and $Q_k = \exists$, respectively.

- $Q_k = \forall$.

  For an arbitrary vector $\mathbf{v} \in G_{k-1}$, we know from lemma 4, that there exists $\mathbf{u} \in G_k$, such that $\mathbf{v} = 2 \cdot \mathbf{u}$, and also $[\mathbf{u}]_k \in G_k$. By induction hypothesis, $G_k \subseteq 2^{n-k} \cdot T_k$ so $\mathbf{u} \in 2^{n-k} \cdot T_k$ and $[\mathbf{u}]_k \in 2^{n-k} \cdot T_k$. By Lemma 2, we know $\mathbf{u} \in 2^{n-k} \cdot T_{k-1}$, or $\mathbf{v} = 2 \cdot \mathbf{u} \in 2^{n-k+1} \cdot T_{k-1}$. Therefore $G_{k-1} \subseteq 2^{n-k+1} \cdot T_{k-1}$.

  On the other hand, For an arbitrary vector $\mathbf{u} \in 2^{n-k+1} \cdot T_{k-1}$, we write $\mathbf{u} = 2 \cdot \mathbf{w}$. Then we know $\mathbf{w} \in 2^{n-k} \cdot T_{k-1}$. Again by Lemma 2, we know both $\mathbf{w}$ and $[\mathbf{w}]_k$ are in $2^{n-k} \cdot T_k$. By induction hypothesis, both $\mathbf{w}$ and $[\mathbf{w}]_k$ are in G_k. So $\mathbf{u} = 2 \cdot \mathbf{w} = \mathbf{w} + [\mathbf{w}]_k \cdot \mathbf{1}_k \in G_{k-1}$.Therefore $G_{k-1} \supseteq 2^{n-k+1} \cdot T_{k-1}$.

- $Q_k = \exists$.
  For an arbitrary vector $\mathbf{v} \in G_{k-1}$, from Lemma 4 we can assume there exists $\mathbf{u} \in G_k$, such that $\mathbf{v} = 2 \cdot \mathbf{u}$ (if it is not the case, consider $[\mathbf{v}]_k$, which is also in G_{k-1}). By induction hypothesis, we have $\mathbf{u} \in 2^{n-k} \cdot T_k$. By Lemma 2, we know both $\mathbf{u} \in 2^{n-k} \cdot T_{k-1}$ and $[\mathbf{u}]_k \in 2^{n-k} \cdot T_{k-1}$, i.e., both $\mathbf{v} = 2 \cdot \mathbf{u} \in 2^{n-k+1} \cdot T_{k-1}$ and $[\mathbf{v}]_k \in 2^{n-k+1} \cdot T_{k-1}$.Therefore $G_{k-1} \subseteq 2^{n-k+1} \cdot T_{k-1}$.
  On the other hand, For an arbitrary vector $\mathbf{u} \in 2^{n-k+1} \cdot T_{k-1}$, we write $\mathbf{u} = 2 \cdot \mathbf{w}$. Then we have $\mathbf{w} \in 2^{n-k} \cdot T_{k-1}$. Again by Lemma 2, we know either $\mathbf{w}$ or $[\mathbf{w}]_k$ is (or both) in $2^{n-k} \cdot T_k$. By induction hypothesis, either $\mathbf{w}$ or $[\mathbf{w}]_k$ is in G_k, and thus either $\mathbf{u} = 2 \cdot \mathbf{w} \in G_{k-1}$ or $\mathbf{u} = 2 \cdot \mathbf{w} \cdot \mathbf{1}_k \in G_{k-1}$.Therefore $G_{k-1} \supseteq 2^{n-k+1} \cdot T_{k-1}$.

■

So if we combine lemma 1 with the All-or-Nothing Rule, we have

Lemma 6 *There exists a polynomial time algorithm A_2, that takes a QBF in standard form, F, and outputs a vector integer circuit C along with the input x, such that F is TRUE, iff $\langle 2^n, 2^n, ..., 2^n \rangle \in C(x)$.*

Proof: The vector integer circuit is constructed as follows: first we apply Lemma 1 to get a circuit that generates the truth table S of the DNF ϕ inside the QBF F. Then we construct another circuit that repeatedly apply the transition function defined in Definition 14 for n times. Putting the two pieces together, we now have a circuit that outputs S_0. By the All-or-Nothing Rule, if F is TRUE, and the filtered set, $G_0 = \psi_0(S_0)$ contains all good vectors, and in particular, vector $\langle 2^n, 2^n, ..., 2^n \rangle$, thus S_0 also contain this vector; if F is FALSE, then G_0 is the empty set, and so S_0 doesn't contain any good vector, nor the vector $\langle 2^n, 2^n, ..., 2^n \rangle$. ■

Notice this result already implies

Theorem 1 *VICP is PSPACE-complete.* ■

3.3 Part 3: Post-processing to extract the result

Now we will reduce the Vector Integer Circuit Problem to the Integer Circuit Problem, where each element in the sets is a positive integer.

We state the Chinese Remainder Theorem first:

Theorem 2 (Chinese Remainder Theorem(CRT)) *For any n positive numbers $a_1, a_2, ..., a_n$ which are pairwise relatively prime, that is, $GCD(a_i, a_j) = 1$, for all $1 \leq i < j \leq n$, let $M := \prod_{i=1}^{n} a_i$. Let*

$$V := \{\mathbf{v} \mid \mathbf{v} \texttt{ is an integer vector and } -a_i/2 < v^i < a_i/2,\ i = 1, 2, ..., n\}$$

Then there exists an 1-1 mapping h from V to $\{1, 2, ..., M\}$. And the mapping is homomorphic, in that for any $\mathbf{u}, \mathbf{v} \in V$, if $\mathbf{u} + \mathbf{v} \in V$ and $\mathbf{u} \cdot \mathbf{v} \in V$, then we have

$$\begin{aligned} h(\mathbf{u}) + h(\mathbf{v}) &\equiv h(\mathbf{u} + \mathbf{v}) \pmod{M} \\ h(\mathbf{u}) \cdot h(\mathbf{v}) &\equiv h(\mathbf{u} \cdot \mathbf{v}) \pmod{M} \end{aligned}$$

■

Now it is almost immediate for us to reduce the VICP problem to ICP problem.

Lemma 7 *There exists a polynomial time algorithm A, which takes a QBF in standard form, F, and output a triple $\langle C, x, N \rangle$, where C is an integer circuit, x is the input to C, and N a positive integer, such that F is TRUE, iff $N \in C(x)$.*

Proof: First by Lemma 6, we have a vector integer circuit C_0, whose output contains $\langle 2^n, 2^n, ..., 2^n \rangle$, iff F is TRUE.

Now we pick the smallest n distinct prime numbers: $p_1 = 2, p_2 = 3, ..., p_n$, and let $a_i := p_i^{n+1}$, for $i = 1, 2, .., n$. Then we know $a_1, a_2, ..., a_n$ are pairwise relatively prime. By the Prime Number Theorem (see [L65]), we know $p_n < n^2$, and thus $a_i = p_i^{n+1} \leq n^{2n} \leq 2^{3n^2}$ and $a_i \geq 2^{n+1}$, for $i = 1, 2..., n$. Let $M = \prod_{i=1}^{n} a_i$. Now notice for each vector generated in circuit C_0, its entry is bounded by 2^n in absolute value (see Lemma 3). So by CRT, we have a 1-1 mapping $h(\cdot)$.

Then we construct an integer circuit C from C_0: for every input gate in C_0 with vector $\mathbf{v}$, we put an input gate in C, with number $h(\mathbf{v})$. for every computational gate in C_0, we put a computational gate in C in the corresponding place. The type of the computational gates in C_0 and C also correspond to each other: for a union gate in C_0, we put a union gate in C; for an addition gate in C_0, we put an addition gate in C; for a multiplication gate in C_0, we put a multiplication gate in C. Finally, we mark the "final" gate in C in the corresponding place of the "final" gate in C_0 — and we denote the output of this "final" gate by R — notice R is a set of positive integers

We let $\alpha := h(\langle 2^n, 2^n, ..., 2^n \rangle)$. Then from CRT and Lemma 6, we know, $\exists m \in R, m \equiv \alpha \pmod{M}$, iff the original QBF, F is TRUE.

Things are becoming a little bit trickier, since we actually doesn't know m — except its residue modulo M. But that turns out not to be a serious problem.

Let $B := R \oplus \{M - \alpha\}$, then we know the original QBF is TRUE, iff B contains a multiple of M.

We will prove (in Lemma 8) that each element in B is bounded by 2^{8n^4}. Now we want to construct a set

$$L := \{k \cdot M \mid 1 \leq k \leq \lambda\}$$

for $\lambda = 2^{8n^4}$.

Then we know B contains a multiple of M, iff

$$(\lambda + 1) \cdot M \in (B \oplus L)$$

We will prove (in Lemma 10) that such a L is constructible using polynomial size integer circuit, and therefore, we can couple the integer circuit computing B and the circuit computing L — we take the addition of them as the final output of the magic circuit and then check if $(\lambda + 1) \cdot M$ is in the output. Then we know $(\lambda + 1) \cdot m \in B \oplus L$ iff the QBF is TRUE. Take N to be $(\lambda + 1) \cdot m$, and we finish the proof.

■

Now we prove that

Lemma 8 *$\|B\| \leq 2^{8n^4}$.*

Proof's sketch: Notice in the construction of the integer circuit C, the numbers in every input gate is bounded by $M = \prod_{i=1}^{n} a_i$, which in turn is bounded by 2^{3n^3}. When we look at the computations within the circuit C, in part 1, constructing the truth table, at most n multiplications (by a constant, which is bounded by M) are performed to each number; in part 2, there are n transition function operations, each has one multiplication with a constant and one addition or a doubling. Therefore the maximum number we can get is bounded by $(2M)^{2n+1} \leq 2^{7n^4}$ — that is a bound for $\|A\|$. Finally since $B := A \oplus \{M - \alpha\}$, we have $\|B\| \leq 2^{8n^4}$.

■

Finally we need to show how to construct L in polynomial time:

Definition 18 *We denote the set $\{1, ..., N\}$ by $[N]$.*

Lemma 9 *For any integer N, we can construct the set $[N]$ using $O(\log N)$ gates.*

Proof's sketch: By induction. Notice if N is odd, then $[N] = ([(N-1)/2] \oplus [(N-1)/2] \oplus \{1\}) \cup \{1, 2\}$; if N is even, then $[N] = ([N/2] \oplus [N/2]) \cup \{1\}$. ■

Lemma 10 *We can construct the set L using $O(n^4)$ gates.*

Proof's sketch: Notice $L = [\lambda] \otimes \{M\}$. ■

So now we have theorem

Theorem 3 *ICP is PSPACE-hard.*

4 Summary and further questions

So we have shown the ICP is PSPACE-hard. This, together with the paper [W84], shows the ICP is PSPACE-complete.

An interesting question about this proof is: the All-or-Nothing Rule is actually stronger than enough for our purpose, and so what we can do about that. One thought is maybe we can weaken the problem a little bit. So consider the "size-version" of the ICP: we still have a circuit C, along with input x and an integer N, but now the problem is not membership but the size — if $|C(x)| = N$ or not.

One can easily modify the Part 3 of the reduction to show the size-version of ICP is also PSPACE-hard, and it is also PSPACE-complete.

But now one can ask the "approximation" problem: given circuit Cm input x, integer N, and a real number ϵ, we ask if N is an approximation of $|C(x)|$ within a factor of ϵ, or, if $(1-\epsilon)|C(x)| \leq N \leq (1+\epsilon)|C(x)|$. Is the problem still PSPACE-complete? Does the complexity depend on ϵ now?

Acknowledgement

The author would like to thank Steven Rudich and Pierre McKenzie for introducing the problem, and especial the latter for the warm encouragement and careful proof-reading of the early manuscript.

References

[AJMV98] E. Allender, J. Jiao, M. Mahajan, V. Vinay, *Noncommutative arithmetic circuits: depth reduction and size lower bounds*, Theoretical computer Science, Vol. 209 (1998), pp. 47 - 86.

[BF91] L. Babai, L. Fortnow, *Arithmetization: A new method in structural complexity theory.* Computational Complexity, 1(1):41-66, 1991.

[BMPT97] M. Beaudry, P. McKenzie, P. Péladeau and D. Thérien, *Finite monoids: from word to circuit evaluation*, Siam J on Computing 2611997138-152

[H82] Lo Keng Hua. *Introduction to Number Theory.* Springer-Verlag, Berlin, 1982.

[CF87] *Jin-Yi Cai and Merrick L. Furst.* PSPACE survives three-bit bottlenecks. In Proceedings, Structure in Complexity Theory, 2nd Annual Conference, pages 94-102, Cornell University, Ithca, NY, 16-19 June 1987. IEEE Computer Society Press.

[K88] E. Kaltofen. *Greatest common divisors of polynomials given by straight-line programs*, J.of ACM, 35:231-264, 1988.

[L65] Calvin T. Long. *Elementary Introduction to Number Theory*, pp 47. D.C.Heath and Compnay, Boston, 1965

[MVW99] Pierre McKenzie, Heribert Vollmer, Klaus W. Wagner. *Arithmetic Circuits and Polynomial Replacement Systems.* Submitted to the same conference.

[P77] D. A. Plaisted, *Sparse complex polynomials and polynomial reducibility*, J. Computer and System Science, 14(1977), pp. 210-221.

[P94] Christos H. Papadimitriou, *Computational Complexity*, pp. 455 - 491. Addison Wesley, 1994.

[S90] A. Shamir, *IP = PSPACE.* 31st FOCS'90, pp 11-15.

[SM73] L. J. Stockmeyer, A. R. Meyer. *Word problems requiring exponential time.* The 5th STOC'73, pp. 1 – 9.

[W84] Klaus W. Wagner. *The complexity of problems concerning graphs with regularities.* Technical report N/84/52, Friedrich-Schiler-Universität Jena, 1984. Extended abstract in *Proceedings of the 11th Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 176, pp. 544 – 552, Springer-Verlag, 1984.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'