A Halting Algorithm to Determine the Existence of Decoder

ShengYu Shen, Ying Qin, JianMin Zhang, and SiKun Li
School of Computer Science, National University of Defense Technology
410073,ChangSha,China
Email: {syshen,qy123,jmzhang,skli}@nudt.edu.cn

***Abstract*—Complementary synthesis automatically synthesizes the decoder circuit E^{-1} of an encoder E. It determines the existence of E^{-1} by checking the parameterized complementary condition (PC). However, this algorithm will not halt if E^{-1} does not exist. To solve this problem, we propose a novel halting algorithm to check PC in two steps.**

First, we over-approximate PC with the linear path unique condition (LP), and then falsify LP by searching for a loop-like path. If such a loop is found, then E^{-1} does not exist; otherwise, LP can eventually be proved within E's recurrence diameter.

Second, with LP proved above, we construct a list of approximations that forms an onion-ring between PC and LP. The existence of E^{-1} can be proved by showing that E belongs to all these rings.

To illustrate its usefulness, we have run our algorithm on several complex encoder circuits, including PCIE and 10G Ethernet. Experimental results show that our new algorithm always distinguishes correct Es from incorrect ones and halts properly.

***Index Terms*—Halting Algorithm, Complementary Synthesis**

I. Introduction

Complementary synthesis has been proposed by us [1] to automatically synthesize an encoder circuit E's decoder E^{-1} in two steps. **First**, it determines the existence of E^{-1} by checking the parameterized complementary condition(***PC***), i.e., whether E's input can be uniquely determined by its output on a bounded unfolding of E's transition function. **Second**, it builds E^{-1} by characterizing its Boolean function with an all-solution SAT solver.

However, the bounded nature of the first step makes it an incomplete algorithm that will not halt if E^{-1} does not exist.

To solve this problem, as shown in Figure 1, we propose a novel halting algorithm to check PC in two steps:

1) First, we over-approximate PC with the linear path unique condition (***LP***), i.e., every linear path of E longer than a particular parameter p always reaches the unique state set S^U, in which the input letter can be uniquely determined by the output letter, the current state and the next state. We then define the negative condition of LP,i.e., the loop-like non-unique condition(***LL***). We can falsify LP and prove LL by searching for a loop-like path that does not reach S^U within E's recurrence diameter rd. If we find such a loop-like path, then LL is proved and E^{-1} does not exist; otherwise, a parameter p can eventually be found to prove LP. In this case, we need the second step below to further check PC.
2) Second, with p found in the first step that proves LP, we construct a list of approximations that forms an onion-ring between PC and its over-approximation LP. If E is found in a certain ring but not in the next inner ring, then PC is falsified and E^{-1} does not exist; otherwise, the existence of E^{-1} is proved.

We have implemented our algorithm with the OCaml language, and solved the generated SAT instances with Zchaff SAT solver [2]. The benchmark set includes several complex encoders from industrial projects (e.g., PCIE and Ethernet), and their slightly modified variants without corresponding decoders. Experimental results show that our new algorithm always distinguishes correct encoders from their incorrect variants and halts properly. All experimental results and programs can be downloaded from http://www.ssypub.org.

This paper's contribution is: We propose the first halting algorithm to determines the existence of an encoder's decoder.

The remainder of this paper is organized as follows. Section II presents background materials. Section IV introduces how to over-approximate PC with LP, and how to falsify LP by searching for loop-like paths. Section V discusses how to construct the onion-ring, and how to determine whether E belongs to a certain ring. Section VI describes how to remove redundant output letters to minimize circuit area, while Section VII and VIII present experimental results and related works. Finally, Section IX concludes with a note on future work.

II. Preliminaries

A. Basic Notation of Propositional Satisfiability Problem

For a Boolean formula F over a variable set V, the **Propositional Satisfiability Problem**(abbreviated as **SAT**) is to find a satisfying assignment $A : V \rightarrow \{0, 1\}$, so that F can

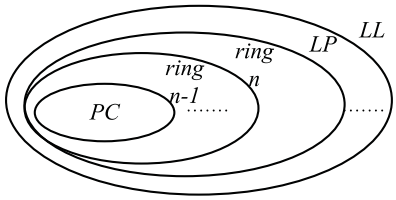


Fig. 1. Relationship between PC,LP and LL

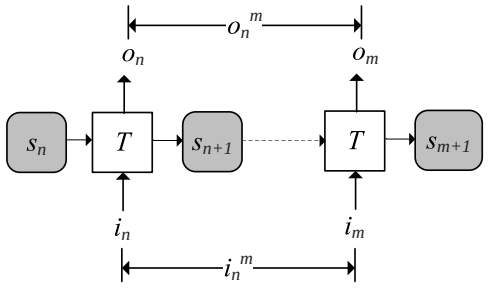


Fig. 2. Mealy finite state machine

be evaluated to 1. If such a satisfying assignment exists, then F is **satisfiable**; otherwise, it is **unsatisfiable**.

A computer program that decides the existence of such a satisfying assignment is called a **SAT solver**, such as Zchaff [2], Grasp [3], Berkmin [4], and MiniSAT [5]. A formula to be solved by a SAT solver is also called a **SAT instance**.

B. Recurrence Diameter

A circuit can be modeled by **Kripke structure** $M = (S, I, T, A, L)$, with a finite state set S, the initial state set $I \subseteq S$, the transition relation $T \subseteq S \times S$, and the labeling of the states $L : S \to 2^A$ with atomic proposition set A.

Kroening et al. [6] defined the **state variables recurrence diameter** with respect to M, denoted by $rrd(M)$, as the longest loop-free path in M starting from an initial state.

$$rrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \dots s_i : I(s_0) \wedge \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\} \quad (1)$$

In this paper, we define a similar concept: the **uninitialized state variables recurrence diameter** with respect to M, denoted by $uirrd(M)$, is the longest loop-free path in M.

$$uirrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \dots s_i : \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\} \quad (2)$$

The only difference between these two definitions is that our $uirrd$ does not consider the initial state.

These definitions are only used in proving our theorems below. Our algorithm does not need to compute these diameters.

C. The Original Algorithm to Determine the Existence of Decoder

The complementary synthesis algorithm [1] includes two steps: determining the existence of decoder and characterizing its Boolean function. We will only introduce the first step here.

The encoder E can be modeled by a Mealy finite state machine [7].

Definition 1: **Mealy finite state machine** is a 5-tuple $M = (S, s_0, I, O, T)$, consisting of a finite state set S, an initial state $s_0 \in S$, a finite set of input letters I, a finite set of output letters O, a transition function $T : S \times I \to S \times O$ that computes the next state and output letter from the current state and input letter.

As shown in Figure 2, as well as in the remainder of this paper, the state is represented as a gray round corner box, and the transition function T is represented by a white rectangle.

We denote the state, input letter and output letter at the n-th cycle respectively as s_n, i_n and o_n. We further denote the sequence of state, input letter and output letter from the n-th to the m-th cycle respectively as s_n^m, i_n^m and o_n^m.

A sufficient condition for the existence of E^{-1} is the **unique condition**, i.e., there exist two parameters d and l, so that i_n of E can be uniquely determined by the output sequence o_{n+d-l}^{n+d-1}. As shown in Figure 3, d is the relative delay between o_{n+d-l}^{n+d-1} and the input letter i_n, while l is the length of o_{n+d-l}^{n+d-1}.

However, the unique condition is unnecessarily restrictive, because it may not hold when s_n is not reachable, even if E is a correct encoder whose input can be uniquely determined by its output in its reachable state set. So we need to rule out unreachable states before checking the unique condition.

The continuous running character of communication circuits provides us an opportunity to rule out unreachable states easily without paying the expensive cost of computing the reachable state set. That is to say, we only need to check the unique condition on the state set RS^∞ that can be reached infinitely often from S.

$$RS^q \stackrel{def}{=} \{s_q | \bigwedge_{m=0}^{q-1} \{(s_{m+1}, o_m) \equiv T(s_m, i_m)\}\} \quad (3)$$

$$RS^{>p} \stackrel{def}{=} \bigcup_{q>p} RS^q \quad (4)$$

$$RS^\infty \stackrel{def}{=} \lim_{p \to \infty} RS^{>p} \quad (5)$$

Here, RS^q is the set of states that can be reached from S with exact q steps.

According to Equation (5) and Figure 3, RS^∞ can be easily over-approximated by prepending a state transition sequence of length p to s_n, which forces s_n to be in the state set $RS^{>p} = \bigcup_{q>p} RS^q$. Obviously, RS^∞ and all $RS^{>p}$ form a total order shown below, which means a tighter over-approximation of RS^∞ can be obtained by increasing the length p of prepended state transition sequence.

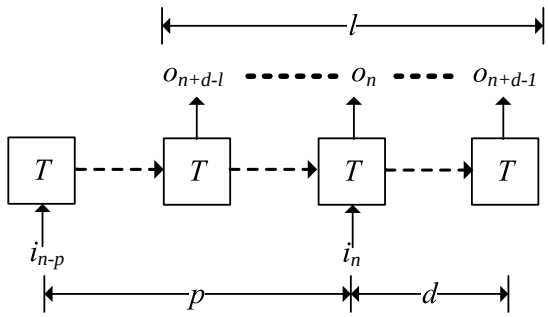


Fig. 3. The parameterized complementary condition

$RS^{\infty} \subseteq \cdots \subseteq RS^{>p2} \subseteq \cdots \subseteq RS^{>p1} \subseteq \ldots$ where $p2 > p1$

Thus, as shown in Figure 3, the parameterized complementary condition(PC) [1] can be defined as:

***Definition** 2:* **Parameterized Complementary Condition (PC) :** For encoder E, $E \vDash PC(p,d,l)$ holds if i_n can be uniquely determined by o_{n+d-l}^{n+d-1} on s_{n-p}^{n+d-1}. This equals the unsatisfiability of $F_{PC}(p,d,l)$ in Equation (6). We further define $E \vDash PC$ as $\exists p,d,l : E \vDash PC(p,d,l)$.

$$F_{PC}(p,d,l) \stackrel{def}{=} \left\{ \begin{array}{c} \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1},o_m) \equiv T(s_m,i_m)\} \\ \wedge \quad \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1},o'_m) \equiv T(s'_m,i'_m)\} \\ \wedge \quad \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\ \wedge \quad i_n \neq i'_n \end{array} \right\} \quad (6)$$

The 2nd and 3rd lines of Equation (6) correspond respectively to two unfolded instances of E's transition function. The only difference between them is that a prime is appended to every variable in the 3rd line. The 4th line forces the output sequences of these two unfolded instances to be the same, while the 5th line forces their input letters to be different.

III. Case Studies

To facilitate the understanding of our idea, we use some small examples shown in Figure 4,5 and 6.

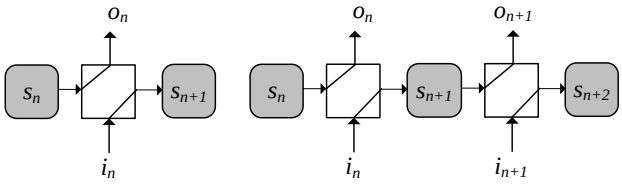
A. Case study 1

The circuit in Figure 4a) stores its input port i_n in register s_{n+1}, and then outputs it to output port o_{n+1}. The unfolding of its transition function is shown in Figure 4b).

Obviously, i_n is same as, and therefore can be uniquely determined by s_{n+1}. So i_n can be uniquely determined by s_n, o_n and s_{n+1}. So LP is satisfied by this circuit. Here, the tuple $< s_n, o_n, s_{n+1} >$ can be seen as a ring that surrounds i_n.

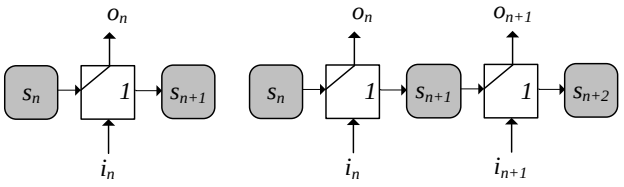
Next, we expand the ring $< s_n, o_n, s_{n+1} >$ to another ring $< s_n, o_n, o_{n+1}, s_{n+2} >$, and perform the following 3 checks:

1) Whether i_n can be uniquely determined by the ring $< s_n, o_n, o_{n+1}, s_{n+2} >$? Obviously the answer is yes.
2) Whether i_n can be uniquely determined by $< o_n, o_{n+1}, s_{n+2} >$, the ring with s_n removed? Obviously the answer is yes.
3) Whether i_n can be uniquely determined by $< o_n, o_{n+1} >$, the ring with s_n and s_{n+2} both removed? Obviously the answer is yes. In this case, we find that i_n can be uniquely determined by the output sequence $< o_n, o_{n+1} >$. Thus, a decoder exists for this circuit.



a) Mealy machine b) Unfolding

Fig. 4. Case study 1



a) Mealy machine b) Unfolding

Fig. 5. Case study 2

B. Case study 2

The circuit in Figure 5a) connects a constant 1, instead of input port i to register s. So i_n can never be determined by s_n, o_n and s_{n+1} in all states. Thus, a loop-like path with length 1 will reach such a state, which satisfies LL and falsifies LP. So no decoder exists for this circuit.

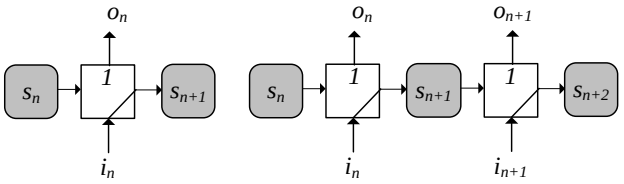
C. Case study 3

For the circuit in Figure 6a), the unfolding of its transition function is shown in Figure 6b). It's output is driven by constant 1, instead of register s. Obviously, this circuit can satisfy LP, which means i_n can be uniquely determined by s_n,o_n and s_{n+1}.

Next, we expand the ring $< s_n, o_n, s_{n+1} >$ to another ring $< s_n, o_n, o_{n+1}, s_{n+2} >$, and perform the following 3 checks:

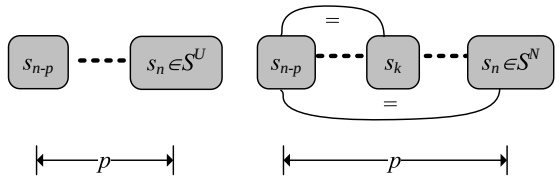
1) Whether i_n can be uniquely determined by the ring $< s_n, o_n, o_{n+1}, s_{n+2} >$? The answer is no, because i_n never goto o_n, o_{n+1} and s_{n+2}.
2) Whether i_n can be uniquely determined by $< o_n, o_{n+1}, s_{n+2} >$, the ring with s_n removed? The answer is still no with the same reason.
3) Whether i_n can be uniquely determined by $< o_n, o_{n+1} >$, the ring with s_n and s_{n+2} both removed? The answer is still no with the same reason.

So in this case, no more expansion is needed, no decoder exists for this circuit.



a) Mealy machine b) Unfolding

Fig. 6. Case study 3



a) The linear path unique condition

b) The loop-like non-unique condition

Fig. 7. Two new uniqueness conditions

IV. Over-approximating PC with LP and Falsifying LP by Searching for Loop-like Path

A. Definition of Over-approximation

We first present some related definitions before defining the over-approximation of PC.

***Definition** 3:* **Unique State Set S^U and Non-unique State Set S^N:** For a circuit E, its unique state set $\boldsymbol{S^U}$ is the set of states s_n that makes i_n to be uniquely determined by s_n,o_n and s_{n+1}, i.e., makes F^U in Equation (7) unsatisfiable. The non-unique state set S^N is the complementary set of S^U, i.e., $\boldsymbol{S^N \stackrel{def}{=} S - S^U}$.

$$F^U \stackrel{def}{=} \left\{ \begin{array}{c} (s_{n+1}, o_n) \equiv T(s_n, i_n) \\ \wedge (s'_{n+1}, o'_n) \equiv T(s'_n, i'_n) \\ \wedge o_n \equiv o'_n \wedge s_n \equiv s'_n \wedge s_{n+1} \equiv s'_{n+1} \\ \wedge i_n \neq i'_n \end{array} \right\} \quad (7)$$

To obtain a halting algorithm, we need to develop a negative condition for PC, which can recognize all those $E \vDash \neg PC$.

Unfortunately, it is very difficult, if not impossible, to develop such a condition. So we choose to first over-approximate PC with the linear path unique condition(LP), and then develop a negative condition for LP, i.e., the loop-like non-unique condition(LL). The definitions of LP and LL are given below, and presented intuitively in Figure 7a) and 7b).

***Definition** 4:* **Linear Path Unique Condition (LP) :** For encoder E, $E \vDash LP(p)$ holds if every linear path of length p always reaches the unique state set S^U. This equals the unsatisfiability of $F_{LP}(p)$ in Equation (8). We further define $E \vDash LP$ as $\exists p : E \vDash LP(p)$.

$$F_{LP}(p) \stackrel{def}{=} F^U \wedge \bigwedge_{m=n-p}^{n} \{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \quad (8)$$

***Definition** 5:* **Loop-like Non-unique Condition (LL) :** For encoder E, $E \vDash LL(p)$ holds if there exists a loop-like path of length p that reaches the non-unique state set S^N. This equals the satisfiability of $F_{LL}(p)$ in Equation (9). We further define $E \vDash LL$ as $\exists p : E \vDash LL(p)$.

$$F_{LL}(p) \stackrel{def}{=} F_{LP}(p) \wedge \bigvee_{m=n-p+1}^{n} \{s_m \equiv s_{n-p}\} \quad (9)$$

Equation (9) is very similar to Equation (8), except that $\bigvee_{m=n-p+1}^{n} \{s_m \equiv s_{n-p}\}$ is inserted to find a loop-like path.

The intuition behind LP and LL is to check whether i_n can be uniquely determined by s_n,s_{n+1} and o_n with prepended s_{n-p}^{n-1}. Here, parameters d and l are removed, which makes it easier to find the value of p .

B. Relationships between PC, LP and LL

The relationships between PC, LP and LL are :

1. LP over-approximates PC,i.e.,$E \vDash PC \rightarrow E \vDash LP$.

2. Between LP and LL, there is always one and only one that holds,i.e.,$E \vDash LP \leftrightarrow E \vDash \neg LL$.

These relationships are presented intuitively in Figure 1, and their proofs are presented below. Those impatient readers can skip the remainder of this subsection.

Before proving these theorems, we need a lemma that defines a new formula for LP.

***Lemma** 1:* With $\overline{F}_{LP}(p, d, l)$ defined below:

$$\overline{F}_{LP}(p, d, l) \stackrel{def}{=} \left\{ \begin{array}{c} \bigwedge_{m=n-p}^{n+d-1} \{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \\ \wedge \quad \bigwedge_{m=n-p}^{n+d-1} \{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\} \\ \wedge o_n \equiv o'_n \wedge s_n \equiv s'_n \wedge s_{n+1} \equiv s'_{n+1} \\ \wedge i_n \neq i'_n \end{array} \right\} \quad (10)$$

we have: $\overline{F}_{LP}(p, d, l) \leftrightarrow F_{LP}(p)$

Proof: **First, for the $\rightarrow$ direction.** It is obvious that the clause set of $\overline{F}_{LP}(p, d, l)$ is a supper set of $F_{LP}(p)$, so the $\rightarrow$ direction is proved.

Second, to prove the $\leftarrow$ direction, we list below all additional sub-formulas that have been added into Equation (8) to obtain (10), and also our methods to satisfy them with a particular satisfying assignment A of $F_{LP}(p)$.

1. $\bigwedge_{m=n-p}^{n} \{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\}$: This formula can be satisfied by assigning $A(s_m)$, $A(i_m)$ and $A(o_m)$ to s'_m, i'_m and o'_m respectively.

2. $\bigwedge_{m=n+1}^{n+d-1} \{(s_{m+1}, o_m) \equiv T(s_m, i_m)\}$: this formula represents a state transition sequence starting from s_{n+1}, which is satisfiable.

3. $\bigwedge_{m=n+1}^{n+d-1} \{(s'_{m+1}\ , o'_m) \equiv T(s'_m, i'_m)\}$: this formula represents a state transition sequence starting from s'_{n+1}, which is satisfiable with the same assignment defined in 2.

So, every satisfying assignment A of $F_{LP}(p)$ can make $\overline{F}_{LP}(p, d, l)$ satisfiable. So the $\leftarrow$ direction is proved.

Thus, this theorem is proved. ■

In the remainder of this paper, we will use $F_{LP}(p)$ and $\overline{F}_{LP}(p, d, l)$ interchangeably.

***Theorem** 1:* $E \vDash PC(p, d, l) \rightarrow E \vDash LP(p)$

Proof: Let's prove it by contradiction. Assume that $A : V \rightarrow \{0, 1\}$ is a satisfying assignment of $\overline{F}_{LP}(p, d, l)$.

We define a new satisfying assignment A' as:

$$A'(v) \stackrel{def}{=} \left\{ \begin{array}{llll} A(o_m) & v \equiv o'_m & m \neq n & \\ A(i_m) & v \equiv i'_m & m \neq n & \\ A(s_m) & v \equiv s'_m & m \neq n & and \quad m \neq n+1 \\ A(v) & otherwise & & \end{array} \right. \quad (11)$$

Thus, A' is also a satisfying assignment of $\overline{F}_{LP}(p, d, l)$.

By comparing Equation (6) with (10), it is obvious that A' is a satisfying assignment of the unsatisfiable formula $F_{PC}(p, d, l)$.

This contradiction concludes the proof. ■

***Theorem** 2:* $E \vDash LP \leftrightarrow E \vDash \neg LL$

Proof: **For the $\rightarrow$ direction**, let's prove it by contradiction. Assume that $E \vDash LL$. This means there exists a loop-like path that reaches state $s_n \in S^N$.

Assume the length of this loop is q, and the parameter of $E \vDash LP$ is p. Then we can unfold this loop $[p/q]+1$ times, to get a path that is longer than p and reaches a state $s_n \in S^N$. This will lead to $E \vDash \neg LP(p)$.

This contradiction concludes the proof of the $\rightarrow$ direction.

For the $\leftarrow$ direction, assume that $E \vDash \neg LP$ and $E \vDash \neg LL$, then for all p, $F_{LP}(p)$ is satisfiable.

Assume the uninitialized state variables recurrence diameter of E is $uirrd$, and let $p = uirrd + 1$. Then $F_{LP}(p)$ is satisfiable, which means there is a path of length p that reaches a state $s_n \in S^N$. Because p is larger than $uirrd$, this path must contain a loop in it, which also makes F_{LL} satisfiable.

So $E \vDash LL$ holds, which contradicts with $E \vDash \neg LL$.

This contradiction concludes the proof of the $\leftarrow$ direction. ■

C. Algorithm to Check $E \vDash LP$ and $E \vDash LL$

Based on the relationships discussed in Subsection IV-B, we develop Algorithm 1(as shown below) to check $E \vDash LP$ and $E \vDash LL$. This algorithm also discovers the value of parameter p if $E \vDash LP$ holds.

Algorithm 1 checkLPLL

```
1: for p = 0 → ∞ do
2:   if F_LP(p) is unsatisfiable then
3:     print "E ⊨ LP(p)"
4:     halt;
5:   else if F_LL(p) is satisfiable then
6:     print "no E^-1 due to E ⊨ LL(p)"
7:     halt;
8:   end if
9: end for
```

According to Theorem 2, Algorithm 1 will eventually halt at line 3 or 6 before p reaches E's uninitialized state variables recurrence diameter $uirrd$. Thus, we have the following theorem.

***Theorem** 3:* Algorithm 1 is a halting algorithm.

With Algorithm 1, we can determine whether E is an improperly designed encoder that leads to $E \vDash LL$. **But if $E \vDash LP$, how to determine whether E is a correct encoder that leads to $E \vDash PC$**? We will discuss this problem in the next section.

V. Checking $E \vDash PC$ by Constructing Onion-Ring

A. Intuitive Description

To make it easier to follow our presentation, we present our idea intuitively here with an example.

As shown in Figure 8, we add two new parameters b and f to replace d and l. The backward parameter b refers to the distance between state s_{n-b} and s_n. The forward parameter f refers to the distance between state s_{n+1} and s_{n+1+f}. The relations between $< b, f >$ and $< d, l >$ are:

$$\begin{aligned} d &= f \\ l &= b + f + 1 \end{aligned} \tag{12}$$

Because Algorithm 1 already recognizes all Es that lead to $E \vDash LL$, we only need to deal with those Es that lead to $E \vDash LP(p)$ here. This will result in the following proposition:

***Proposition** 1:* i_n is uniquely determined by s_n, o_n and s_{n+1}.

As shown in Figure 8, we can further generalize Proposition 1 by:

1) Replacing o_n with o_{n-b}^{n+f},
2) Replacing s_n with s_{n-b},
3) Replacing s_{n+1} with s_{n+f+1},

and thus obtain:

***Proposition** 2:* i_n is uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1}.

It is obvious that Proposition 1 is a special case of Proposition 2, with $b \equiv 0$ and $f \equiv 0$.

With this generalization, our algorithm will be intuitively described in the following five steps:

1) First, we ignore both s_{n-b} and s_{n+f+1}, and test whether i_n can be uniquely determined by o_{n-b}^{n+f}. **If yes, our algorithm halts with $E \vDash PC$.**
2) Otherwise, we ignore s_{n-b}, and test whether i_n can be uniquely determined by o_{n-b}^{n+f} and s_{n+f+1}. If yes, then i_n definitely does **NOT** depend on any o_k with $k < n-b$, but it may still depend on some o_k with $k > n+f$. So we need to increase f by 1 and goto step 1.
3) Otherwise, we ignore s_{n+f+1}, and test whether i_n can be uniquely determined by s_{n-b} and o_{n-b}^{n+f}. If yes, then i_n definitely does **NOT** depend on any o_k with $k > n+f$, but it may still depend on some o_k with $k < n-b$. So we need to increase b by 1 and goto step 1.
4) Otherwise, we test whether i_n can be uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1}. If yes, then i_n may depend on some o_k with both $k < n-b$ and $k > n+f$, so we need to increase b and f by 1, and goto step 1.
5) If the algorithm reaches here, then i_n had been uniquely determined by $s_{n-b'}$, $o_{n-b'}^{n+f'}$ and $s_{n+f'+1}$ previously, but **NOT** by s_{n-b}, o_{n-b}^{n+f} and s_{n+f+1} now, where $b' \leq b$ and $f' \leq f$. This means that adding more o_k into o_{n-b}^{n+f} by

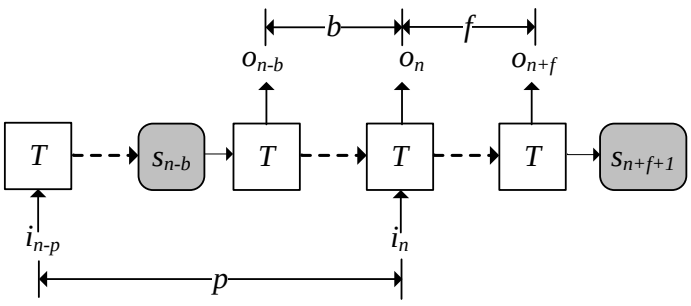


Fig. 8. Forward and backward constraints

increasing b and f, will never make PC holds. **Our algorithm halts with** $E \vDash \neg PC$.

In this algorithm, every step with a pair of b and f corresponds to an onion-ring defined in subsection V-B. If it halts at step 5, then E belongs to the ring corresponding to b' and f', but does not belong to the next inner ring corresponding to b and f, which means E^{-1} does not exist.

Formal presentation and proof will be given in the next two subsections.

B. Constructing Onion-Ring between PC and LP

According to Figure 8, we define the following formulas:

F_{unfold} defines two unfolded instances of transition function, and constrains that their output sequence are equivalent, whereas their input letters are inequivalent:

$$F_{unfold}(p,b,f) \stackrel{def}{=} \left\{ \begin{array}{c} \bigwedge_{m=n-p}^{n+f}\{(s_{m+1},o_m) \equiv T(s_m,i_m)\} \\ \wedge \quad \bigwedge_{m=n-p}^{n+f}\{(s'_{m+1},o'_m) \equiv T(s'_m,i'_m)\} \\ \bigwedge_{m=n-b}^{n+f}\{o_m \equiv o'_m\} \\ \wedge \quad i_n \neq i'_n \end{array} \right\} \tag{13}$$

$F_{backward}$ constrains that s_{n-b} equals s'_{n-b}:

$$F_{backward}(p,b,f) \stackrel{def}{=} \{s_{n-b} \equiv s'_{n-b}\} \tag{14}$$

$F_{forward}$ constrains that s_{n+1+f} equals s'_{n+1+f}:

$$F_{forward}(p,b,f) \stackrel{def}{=} \{s_{n+1+f} \equiv s'_{n+1+f}\} \tag{15}$$

With these formulas, we define 4 new unique conditions between PC and LP.

$\boldsymbol{LP\_nobf(p,b,f)}$: i_n can be uniquely determined by o_{n-b}^{n+f}. This equals the unsatisfiability of F_{LP_nobf} in Equation (16).

$$F_{LP\_nobf}(p,b,f) \stackrel{def}{=} F_{unfold} \tag{16}$$

$\boldsymbol{LP\_f(p,b,f)}$: i_n can be uniquely determined by o_{n-b}^{n+f} and s_{n+1+f}. This equals the unsatisfiability F_{LP_f} in Equation (17).

$$F_{LP\_f}(p,b,f) \stackrel{def}{=} F_{unfold} \wedge F_{forward} \tag{17}$$

$\boldsymbol{LP\_b(p,b,f)}$: i_n can be uniquely determined by s_{n-b} and o_{n-b}^{n+f}. This equals the unsatisfiability F_{LP_b} in Equation (18).

$$F_{LP\_b}(p,b,f) \stackrel{def}{=} F_{unfold} \wedge F_{backward} \tag{18}$$

$\boldsymbol{LP\_bf(p,b,f)}$: i_n can be uniquely determined by s_{n-b}, o_{n-b}^{n+f} and s_{n+1+f}. This equals the unsatisfiability F_{LP_bf} in Equation (19).

$$F_{LP\_bf}(p,b,f) \stackrel{def}{=} F_{unfold} \wedge F_{backward} \wedge F_{forward} \tag{19}$$

These new unique conditions, when coupled with parameters b and f, will act as onion-rings between PC and its over-approximation LP.

It is obvious that, LP_bf is very similar to LP, while LP_nobf is very similar to PC. Such similarities will be employed to prove the correctness of our approach.

Some lemmas that will bed used to prove the correctness of the onion-ring approach are given below:

***Lemma** 2:* $E \vDash LP\_bf(p,b,f) \leftarrow E \vDash LP\_bf(p,b,f+1)$

Proof: Let's prove it by contradiction. Assume that $E \vDash \neg LP\_bf(p,b,f)$ and $E \vDash LP\_bf(p,b,f+1)$, which means that $F_{LP\_bf}(p,b,f)$ is satisfiable while $F_{LP\_bf}(p,b,f+1)$ is unsatisfiable.

We can append a state transition to $F_{LP\_bf}(p,b,f)$ after s_{n+f+1}, and get a new formula $F'_{LP\_bf}(p,b,f)$.

$$\begin{array}{l} F'_{LP\_bf}(p,b,f) \stackrel{def}{=} F_{LP\_bf}(p,b,f) \\ \wedge\,(s_{n+f+2},o_{n+f+1}) = T(s_{n+f+1},i_{n+f+1}) \end{array} \tag{20}$$

This newly appended state transition is satisfiable, which makes F'_{LP_bf} a satisfiable formula.

Assume A is a satisfying assignment of $F'_{LP\_bf}(p,b,f)$. We define another satisfying assignment A' as

$$A'(v) \stackrel{def}{=} \left\{ \begin{array}{ll} A(o_{n+f+1}) & v \equiv o'_{n+f+1} \\ A(i_{n+f+1}) & v \equiv i'_{n+f+1} \\ A(s_{n+f+2}) & v \equiv s'_{n+f+2} \\ A(v) & otherwise \end{array} \right. \tag{21}$$

Obviously, A' is a satisfying assignment of unsatisfiable formula $F_{LP\_bf}(p,b,f+1)$.

This contradiction concludes the proof. ■

***Lemma** 3:* $E \vDash LP\_b(p,b,f) \leftarrow E \vDash LP\_b(p+1,b+1,f)$

Its proof is similar to that of Lemma 2.

***Lemma** 4:* If $E \vDash PC(p,d,l)$, then the following Equation holds.

$$\begin{array}{rlll} & E \vDash & LP\_bf & (p+d-l+1,0,d) \\ \leftrightarrow & E \vDash & LP\_b & (p+d-l+1,0,d) \end{array} \tag{22}$$

Proof: **For the $\leftarrow$ direction**, according to Equation (18) and (19), it is obvious that the clause set of F_{LP_bf} is a super set of F_{LP_b}, which means the unsatisfiability of the latter one implies the unsatisfiability of the former one. So the $\leftarrow$ direction is proved.

For the $\rightarrow$ direction, let's prove it by contradiction. Assume that $F_{LP\_bf}(p+d-l+1,0,d)$ is unsatisfiable, and A is a satisfying assignment of $F_{LP\_b}(p+d-l+1,0,d)$.

We define another assignment A' as :

$$A'(v) \stackrel{def}{=} \left\{ \begin{array}{llll} A(o_k) & v \equiv o'_k & where & k<n \\ A(i_k) & v \equiv i'_k & where & k<n \\ A(s_k) & v \equiv s'_k & where & k<n \\ A(v) & otherwise & & \end{array} \right. \tag{23}$$

Obviously, A' is a satisfying assignment of Equation (6), which means $E \nvDash PC(p,d,l)$. This contradiction concludes the proof of the $\rightarrow$ direction.

■

***Lemma** 5:* If $E \vDash PC(p,d,l)$, then the following Equation holds.

$$\begin{array}{rlll} & E \vDash & LP\_b & (p,l-d-1,d) \\ \leftrightarrow & E \vDash & LP\_nobf & (p,l-d-1,d) \end{array} \tag{24}$$

Its proof is similar to that of Lemma 4.

An important theorem that defines the onion-ring is presented and proved below:

***Theorem** 4:* If $E \vDash PC(p,d,l)$, then there exists a list of unique conditions with their relationship shown below:

$$\begin{array}{cccl} & E \vDash & LP & (p+d-l+1) \\ \leftrightarrow & E \vDash & LP\_bf & (p+d-l+1,0,0) \\ \leftarrow & E \vDash & LP\_bf & (p+d-l+1,0,1) \\ & & \ldots & \\ \leftarrow & E \vDash & LP\_bf & (p+d-l+1,0,d-1) \\ \leftarrow & E \vDash & LP\_bf & (p+d-l+1,0,d) \\ \leftrightarrow & E \vDash & LP\_b & (p+d-l+1,0,d) \\ \leftarrow & E \vDash & LP\_b & (p+d-l+2,1,d) \\ & & \ldots & \\ \leftarrow & E \vDash & LP\_b & (p,l-d-1,d) \\ \leftrightarrow & E \vDash & LP\_nobf & (p,l-d-1,d) \\ \leftrightarrow & E \vDash & PC & (p,d,l) \end{array} \quad (25)$$

Proof: According to Equation (10) and (19), the $\leftrightarrow$ relation between the 1st and 2nd line of Equation (25) holds.

According to Lemma 2, the $\leftarrow$ relations between the 2nd and 6th line of Equation (25) holds.

According to Lemma 4, the $\leftrightarrow$ relation between the 6th and 7th line of Equation (25) holds.

According to Lemma 3, the $\leftarrow$ relations between the 7th and 10th line of Equation (25) holds.

According to Lemma 5, the $\leftrightarrow$ relation between the 10th and 11th line of Equation (25) holds.

According to Equation (6) and (16), the $\leftrightarrow$ relations between the last two lines of Equation (25) holds. ■

In Equation (25), all $\leftarrow$ symbols form a total order, which makes all unique conditions on the right-hand side of $\vDash$s to form an onion-ring(as shown in Figure 1).

C. Algorithm Implementation

With those theorems presented in Subsection V-B, we use the following Algorithm 2 to check $E \vDash PC$.

Algorithm 2 $checkPCLP(p,b,f)$

```
1: if F_{LP_nobf}(p,b,f) is unsatisfiable then
2:    print "E ⊨ PC(p,f,b+f+1)"
3:    halt;
4: else if F_{LP_f}(p,b,f) is unsatisfiable then
5:    checkPCLP(p,b,f+1)
6: else if F_{LP_b}(p,b,f) is unsatisfiable then
7:    checkPCLP(p+1,b+1,f)
8: else if F_{LP_bf}(p,b,f) is unsatisfiable then
9:    checkPCLP(p+1,b+1,f+1)
10: else
11:   print "no E^{-1} due to E ⊨ ¬PC"
12:   halt;
13: end if
```

Algorithm 2 is invoked with the form $checkPCLP(p,0,0)$, with p computed by Algorithm 1.

Algorithm 2 just follows the onion-ring defined by Equation (25), from the first line to the last line. If $E \vDash PC$ holds, it will eventually reach line 2, and the existence of E^{-1} is proved; otherwise, it will eventually reach line 11, which means that E does not belong to the current ring, and PC is falsified. So E^{-1} does not exist.

Thus, with Theorem 4, we have the following theorem.

***Theorem** 5:* Algorithm 2 is a halting algorithm.

VI. Removing Redundant Output Letters

Although Algorithm 1 and 2 together are sufficient to determine the existence of E^{-1}, the parameters found by line 2 of Algorithm 2 contain some redundancy, which will cause unnecessary large overhead of circuit area.

For example, as shown in Figure 9, assume that l is the smallest parameter value that leads to $E \vDash PC(p,d,l)$, and $l < d$, which means that i_n is uniquely determined by some output letters o_k with $k > n$.

We further assume that line 2 of Algorithm 2 find out $E \vDash PC(p,d,l')$. It is obvious that $l' > d$, which make i_n to depend on some redundant o_k with $k \leq n$.

So $o_{n+d-l'}^{n+d-l-1}$ is the sequence of redundant output letters, which should be removed to prevent them from being instantiated as latches in circuit E^{-1}.

Algorithm 3 that removes these redundant output letters is presented below:

Algorithm 3 $RemoveRedundancy(p,d,l')$

```
1: for l = 0 → l' do
2:    if F_PC(p,d,l) is unsatisfiable then
3:       print "E ⊨ PC(p,d,l)"
4:       halt;
5:    end if
6: end for
```

VII. Experimental Results

We have implemented our algorithm in Zchaff [2], and run it on a PC with a 2.4GHz Intel Core 2 Q6600 processor, 8GB memory and CentOS 5.2 linux. All experimental results and programs can be downloaded from http://www.ssypub.org.

A. Benchmarks

Table I shows information of the following benchmarks.

1) A XGXS encoder compliant to clause 48 of IEEE-802.3ae 2002 standard [8].

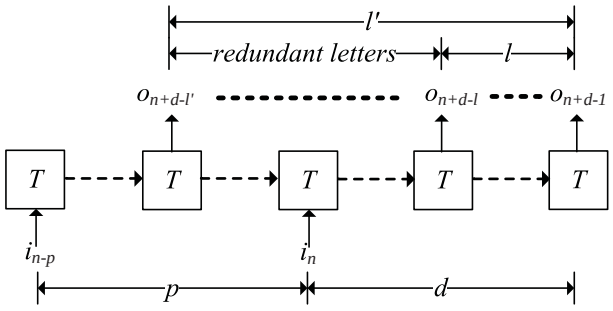


Fig. 9. Redundant Output Letters

TABLE I
INFORMATION OF BENCHMARKS

	XGXS	XFI	scrambler	PCIE	T2 ethernet
Line number of verilog source code	214	466	24	1139	1073
#regs	15	135	58	22	48
Data path width	8	64	66	10	10

2) A XFI encoder compliant to clause 49 of the same IEEE standard.
3) A 66-bit scrambler used to ensure that a data sequence has sufficiently many 0-1 transitions , so that it can run through high-speed noisy serial transmission channel.
4) A PCIE physical coding module.
5) The Ethernet module of Sun's OpenSparc T2 processor.

B. Experimental Results on Properly Designed Encoders

The 2nd and 6th rows of Table II compares the run time of checking $E \models PC$ between [1] and our approach. The run time of our approach are much larger than [1]. This is caused by checking the unique and non-unique conditions defined in Section IV and V.

The 3rd and 7th rows compare the discovered parameter values, and some minor differences are found on parameter p. This is caused by the different orders in checking various parameter combinations.

According to [9], p is used to constrain the reachable states, while d and l will affect the run time of building E^{-1} and its circuit area. To prove this, we compared the run time of building E^{-1} with all-solution SAT solver in the 4th and 8th rows of Table II, and also compared the area of E^{-1} in the 5th and 9th rows. These E^{-1}s were synthesized with DesignCompiler and LSI10 target library.

The results indicate that the differences in parameter p do not cause significant overhead in the run time of all-solution SAT solver and circuit area.

C. Experimental Results on Improperly Designed Encoders

To further show the usefulness of our algorithm, we need some improperly designed encoders without corresponding decoders.

TABLE II
EXPERIMENTAL RESULTS ON PROPERLY DESIGNED ENCODERS

		XGXS	XFI	scrambler	PCIE	T2 ethernet
[1]	time chk PC(sec)	0.49	59.19	2.52	1.46	35.17
	d, p, l	1,0,1	0,3,2	0,1,2	2,1,1	4,0,1
	run time allsat(sec)	1.16	1047.19	2.00	0.96	29.51
	area	765	19443	1455	398	648
Ours	time chk PC(sec)	1.32	88.68	7.23	2.73	84.47
	d, p, l	1,1,1	0,3,2	0,2,2	2,1,1	4,1,1
	run time allsat(sec)	1.38	1055.64	3.23	1.18	29.42
	area	773	19481	1455	400	535

TABLE III
EXPERIMENTAL RESULTS ON IMPROPERLY DESIGNED ENCODERS

	XGXS	XFI	scrambler	PCIE	T2 ethernet
Alg 1 result	$LP(1)$	$LL(2)$	$LL(2)$	$LP(1)$	$LP(1)$
Alg 2 result	$\neg PC$	NA	NA	$\neg PC$	$\neg PC$
time(sec)	1.23	44.58	3.26	1.67	21.49

We obtained these improperly designed encoders by modifying each benchmark's output statements, such that they can explicitly output the same letter for two different input letters. In this way, input letter i_n can never be uniquely determined by E's output sequence.

The 2nd row of Table III shows the result of Algorithm 1, while the 3rd row shows the result of Algorithm 2. The total run time is shown in the 4th row.

For XFI and scrambler, the result of Algorithm 1 is LL, which falsifies PC directly. So the result of Algorithm 2 is **NA**.

The results indicate that our algorithm always terminated, and recognized these modified incorrect encoders.

VIII. RELATED WORKS

A. Complementary Synthesis

The concept of complementary synthesis was first proposed by us [1] in ICCAD 2009. Its major shortcomings are that it is incomplete, and its run-time overhead of building complementary circuit is too large.

The incomplete problem has been addressed by this paper, while we [9] addresses the second shortcoming by simplifying the SAT instance with unsatisfiable core extraction before building complementary circuits.

B. The Completeness of Bounded Model Checking

Bounded model checking(BMC) is a model checking technology that considers only those paths of limited length. Many researchers try to find out complete approaches for BMC.

One line of research [6], [10] tries to find out a bound b, which can guarantee the correctness of a specification on all paths, if the specification is correct on all paths shorter than b.

The other line of research [11] tries to find out a pattern for induction, such that the correctness of a specification within any bound b implies the correctness on bound $b+1$.

Our approach achieves completeness without following these two approaches. Instead, we define two complement uniqueness conditions, LP and LL, and find out proper algorithms to check them.

C. Temporal Logic Synthesis

The temporal logic synthesis was first addressed by Clarke et.al [12] and Manna et.al [13]. But Pnueli et.al [14] pointed out that the complexity of LTL synthesis is double exponent.

One line of research [15]–[17] focuses on the so-called generalized reactive formulas of the form: $(\Box\Diamond p_1 \wedge \cdots \Box\Diamond p_m) \rightarrow (\Box\Diamond q_1 \wedge \cdots \Box\Diamond q_n)$. Complexity of solving synthesis problem for such formula is $O(N^3)$.

The other line of research focuses on finding efficient algorithm [18] for expensive safra determination algorithm [19] on an useful formula subset, or just avoiding it [20].

Based on these research works, some tools [21] that can handle small temporal formulas have been developed.

All these works assume a hostile environment, which seems too restrictive for many applications. So Fisman et.al [22], Chatterjee et.al [23] and Ummels et.al [24] proposed rational synthesis algorithm, which assumes that each agents act to achieve their own goals instead of failing each other.

D. Protocol Converter Synthesis

The protocol converter synthesis was first proposed by Avnit et.al [25] to automatically generate a translator between two different communication protocols. Avnit et.al [26] improved it with a more efficient design space exploration algorithm. The implementation of this tool is introduced in [27].

IX. Conclusions and Future Works

This paper proposes the first halting algorithm that checks whether a particular encoder E has corresponding decoder. Theoretical analysis and experimental results show that our approach always distinguishes correct encoders from their incorrect variants and halts properly.

One future work is to develop a debugging method to find out why E^{-1} does not exist. For the failure caused by loop-like path, we plan to develop a debugging mechanism based on our previous work on loop-like counterexample minimization [28].

Acknowledgment

The authors would like to thank the editors and anonymous reviewers for their hard work.

This work was fund by project 60603088 supported by National Natural Science Foundation of China.

This work was also supported by the Program for Changjiang Scholars and Innovative Research Team in University No IRT0614.

References

[1] ShengYu Shen, JianMin Zhang, Ying Qin, SiKun Li. Synthesizing Complementary Circuits Automatically. in ICCAD'09, pp 381-388, 2009.

[2] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: Engineering an Efficient SAT Solver. In DAC'01, pp 530-535, 2001.

[3] João P. Marques Silva, Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. in ICCAD'96, pp 220-227, 1996.

[4] E. Goldberg, Y Novikov. BerkMin: A Fast and Robust Sat-Solver. in DATE'02, pp 142-149, 2002.

[5] N. Eén, N. Sörensson. Extensible SAT-solver. in SAT'03, pp 502-518, 2003.

[6] D. Kroening, Ofer Strichman. Efficient Computation of Recurrence Diameters. in VMCAI'03, pp 298-309, 2003.

[7] Mealy, George H. A Method for Synthesizing Sequential Circuits. Bell Systems Technical Journal v 34, pp 1045-1079, 1955.

[8] *IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation*, IEEE Std. 802.3, 2002.

[9] ShengYu Shen, Ying Qin, KeFei Wang, LiQuan Xiao, JianMin Zhang, SiKun Li. Synthesizing Complementary Circuits Automatically. IEEE transaction on CAD of Integrated Circuits and Systems, 29(8):1191-1202, 2010.

[10] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Yunshan Zhu. Symbolic Model Checking without BDDs. In TACAS'99, pp 193-207, 1999.

[11] Mary Sheeran, Satnam Singh, Gunnar Stalmarck. Checking Safety Properties Using Induction and a SAT-Solver. In FMCAD'00, pp 108-125, 2000.

[12] E.M. Clarke, E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In IBM Workshop on Logics of Programs,LNCS 131, pp 52-71, 1981.

[13] Z. Manna, P. Wolper. Synthesis of communicating processes from temporal logic specifications. ACM Trans. Prog. Lang. Sys., 6:68-93, 1984.

[14] A. Pnueli, R. Rosner. On the synthesis of a reactive module. In Proc. 16th ACM Symp. Princ. of Prog. Lang.,pages 179-190, 1989.

[15] E. Asarin, O. Maler, A. Pnueli, J. Sifakis. Controller synthesis for timed automata. In IFAC Symposium on System Structure and Control, pages 469-474. Elsevier, 1998.

[16] R. Alur, S. La Torre. Deterministic generators and games for LTL fragments. ACM Trans. Comput. Log., 5(1):1-25,2004.

[17] N. Piterman, A. Pnueli, Y. Saar, Synthesis of Reactive(1) Designs, in VMCAI'06, pp 364-380, 2006.

[18] O. Maler, D. Nickovic, A. Pnueli. On Synthesizing Controllers from Bounded-Response Properties. In CAV'07, pp 95-107, 2007.

[19] S. Safra. Complexity of Automata on Infinite Objects. PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, March 1989.

[20] A. Harding, M. Ryan, P. Schobbens. A New Algorithm for Strategy Synthesis in LTL Games. in TACAS'05, pp 477-492, 2005.

[21] B. Jobstmann, S. Galler, M. Weiglhofer, R. Bloem. Anzu: A Tool for Property Synthesis. in CAV'07, pp 258-262, 2007.

[22] D Fisman, O Kupferman, Yoad Lustig. Rational Synthesis. in TACAS'10, pp 190-204, 2010.

[23] Chatterjee, K., Henzinger, T.A. Assume-guarantee synthesis. In TACAS'07, pp 261-275, 2007.

[24] Ummels, M. Rational behaviour and strategy construction in infinite multiplayer games. In FSTTCS'06, pp 212-223, 2006.

[25] K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, S. Parameswaran. A Formal Approach To The Protocol Converter Problem. in DATE'08, pp 294-299, 2008.

[26] K. Avnit, A. Sowmya. A formal approach to design space exploration of protocol converters. in DATE'09, pp 129-134, 2009.

[27] K. Avnit, A. Sowmya, J. Peddersen. ACS: Automatic Converter Synthesis for SoC Bus Protocols. in TACAS'10, pp 343-348, 2010.

[28] ShengYu Shen, Ying Qin, Sikun Li. Minimizing Counterexample of ACTL Property. in CHARME'05, pp 393-397, 2005.