Efficient Computation of Observer Projections using OP-Verifiers

P. N. Pena∗ J. E. R. Cury∗∗ R. Malik∗∗∗ S. Lafortune****

* Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerais, Brasil (e-mail: ppena@ufmg.br)
** Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Brasil (e-mail: cury@das.ufsc.br)
*** Department of Computer Science, The University of Waikato, New Zealand, (e-mail: robi@cs.waikato.ac.nz)
**** Department of Electrical Engineering and Computer Science, The University of Michigan, USA (e-mail: stephane@eecs.umich.edu)

Abstract: This paper proposes a procedure to compute abstractions of Discrete Event System (DES) models with the observer property (OP). The procedure, named OP-Search, is based on the OP-Verifier algorithm which verifies if a given natural projection has the observer property. In case OP fails for a projection in Σ of an automaton M, OP-Search modifies M by relabelling transitions and incorporating the new events in Σ, in a way that the modified natural projection generates OP-abstractions. Although OP-Search does not guarantee minimal abstracted models, it leads in general to very reasonable solutions and is shown to be of lower time complexity compared to previous work in the literature.

Keywords: Finite state machines, Discrete event systems, Supervisory control, Abstraction, Model Reduction, Observers, Nonblocking.

1. INTRODUCTION

Computing abstractions for Discrete Event Systems (DES) models is an extensively used approach to deal with the state space explosion inherent to this class of dynamic systems. Two particular DES approaches consider the use of abstractions, the hierarchical and modular control architectures. The observer property (OP) is an important condition to be satisfied by the abstracted models used in both approaches. It was first introduced in the context of hierarchical control (Wong and Wonham, 1996), where the abstracted model is used to represent the high-level of the control hierarchy. In (Wong and Wonham, 1996), the abstraction is obtained in the form of a reporter map, which projects strings of events of the original (low-level) model, built from a set Σ, into high-level strings built from an independent set T of events.

Unfortunately, reporter maps are difficult to interpret as abstractions (Feng and Wonham, 2010), and most applications focus on abstracted models obtained by the so-called natural projection, which simply erases certain events from the low-level model to give the abstracted high-level model. Observers obtained by natural projections have been used in hierarchical control (Hill and Tilbury, 2006; Cunha and Cury, 2007; Schmidt and Breindl, 2008; Feng and Wonham, 2008; Schmidt et al., 2008), in modular synthesis (Hill and Tilbury, 2006; Schmidt et al., 2006; Feng and Wonham, 2006), and in compositional verification of the nonblocking property (Pena et al., 2009) of discrete event systems.

In (Wong and Wonham, 2004), a polynomial procedure is proposed to obtain optimal OP reporter maps, by modifying an initial given map where OP fails. In (Feng and Wonham, 2010) it is shown that the problem of obtaining minimal natural observers by extending a given set of events is NP-hard, and a polynomial algorithm is presented to find reasonable extensions of events leading to natural observers.

Recently, an algorithm called OP-verify was introduced in (Pena et al., 2008) to test if an abstraction obtained by natural projection has the observer property. The OP-Verify algorithm was inspired by the verifier introduced in (Yoo and Lafortune, 2002). In this paper, an OP-Search procedure based on the OP-verify algorithm is proposed to compute, in case OP fails for a given set of events, OP-abstractions by appropriately renaming non-relevant transitions and incorporating them in the relevant set of events. This way of modifying natural projections is suitable for many control problems, as discussed in the paper. Although OP-Search does not guarantee minimal extensions, the way transitions are chosen leads in general to very reasonable natural observers. Also, it is shown in the paper that the proposed procedure has computational gains when compared to the above cited works.

This paper is organised as follows. Section 2 introduces the necessary background, and Section 3 introduces the OP-verify from (Pena et al., 2008) with its properties. Then, Section 4 describes the OP-search algorithm to find
observer projections, with proofs of correctness and termination, and a discussion of complexity. Then, Section 5 demonstrates the algorithm’s execution using an example, and Section 6 adds some concluding remarks.

2. PRELIMINARIES

This paper is set in the supervisory control framework initiated by (Ramadge and Wonham, 1989). The reader is referred to (Cassandras and Lafortune, 2007) for a detailed introduction to the theory.

Discrete system behaviours are modelled using traces of events taken from a finite alphabet Σ. Then Σ∗ is the set of all finite traces of events in Σ, including the empty trace ε. The concatenation of traces s, u ∈ Σ∗ is written as su. A trace s ∈ Σ∗ is called a prefix of t ∈ Σ∗, written s ⊆ t, if there exists u ∈ Σ∗ such that su = t. A subset L ⊆ Σ∗ is called a language. The prefix-closure τ of a language L ⊆ Σ∗ is the set of all prefixes of traces in L, i.e., τ = {s ∈ Σ∗ | s ⊆ t for some t ∈ L}.

For Σ, ⊆ Σ, the natural projection θ: Σ∗ → Σ∗ maps traces in Σ∗ to traces in Σ∗ by erasing all events not contained in Σ. The concept is extended to languages by defining θ(L) = {t ∈ Σ∗ | t = θ(s) for some s ∈ L}.

This paper is concerned about the property of projections known as the observer property, which is introduced in (Wong and Wonham, 1996) for prefix-closed languages and extended to general languages in (Wong et al., 2000).

Definition 1. (Wong et al., 2000) Let L ⊆ Σ∗ be a language, let Σ, ⊆ Σ, and let θ: Σ∗ → Σ∗ be the natural projection. If for all s ∈ τ and all t ∈ Σ∗ such that θ(s)t ∈ θ(L), there exists t′ ∈ Σ∗ such that θ(st′) = θ(s)t and st′ ∈ L, then θ(L) has the observer property.

Discrete event systems are modelled as automata M = (Q,M, Σ, δ,M, q0,M), where Q,M is the set of states, Σ is the alphabet of events, δ,M: Q × Σ → Q,M is the (partial) transition function, and q0,M ∈ Q,M is the initial state. The notation δ,M(x, σ) means that the function δ,M is defined for inputs x ∈ Q,M and σ ∈ Σ. For x ∈ Q, the set of enabled events at state x of M is En,M(x) = {σ ∈ Σ | δ,M(x, σ) exists}. The behaviour of M, modelled as a language L(M) ⊆ Σ∗, is the set of all finite traces that M can generate.

To express termination, the alphabet Σ is assumed to contain the special event τ ∈ Σ, which may only appear on selfloops, i.e., δ,M(x, τ) = x whenever δ,M(x, τ) exists. In this notation, the marked behaviour of M is defined as L(M) = {s ∈ (Σ − {τ})∗ | sτ ∈ L(M)}.

This paper uses the termination event in favour of the more conventional set of terminal states, because it simplifies presentation by unifying termination with ordinary events. A traditional automaton with terminal states, G = (Q,G, Σ,G, δ,G, q0,G, F,G) with τ ∉ Σ,G, can be converted into this paper’s M = (Q,M, Σ, δ,M, q0,M) by adding τ to the alphabet, Σ = Σ,G∪{τ}, letting Q,M = Q,G and q0,M = q0,G, and adding τ-selfloops to every terminal state in F,G, i.e., δ,M(x, σ) = δ,G(x, σ) for σ ≠ τ and δ,M(x, τ) = x for all x ∈ F,G.

Projection can also be applied to automata, and in the following, it will be said that θ(M) has the observer property if θ(L(M)) has the observer property. In this case θ(M) is also called an OP-abstraction.

The observer property is of great interest in compositional verification (Flordal and Malik, 2009) and modular synthesis (Feng and Wonham, 2006; Hill and Tilbury, 2006) of large-scale discrete event systems. When considering a system composed of several automata M1, M2, . . . , Mn, the compositional and modular approach exploit the fact that some components Mi typically use local events, i.e., events not shared with any other automata Mj, j ≠ i. As these events are not used in synchronisation, it is desirable to remove them in order to simplify the problem. By the results of (Pena et al., 2009; Feng and Wonham, 2006), natural projection can be used to erase such local events, provided that the observer property is satisfied.

3. OP-VERIFIER AND SOME PROPERTIES

The observer computation algorithm proposed in this paper is based on the OP-Verifier algorithm introduced in (Pena et al., 2008), which tests if an abstraction obtained through natural projection has the observer property. In the following, the main concepts of the OP-verifier algorithm are introduced, and some additional properties needed in later sections are proved.

3.1 The OP-Verifier Algorithm

The OP-Verifier algorithm was inspired by the verifier introduced in (Yoo and Lafortune, 2002). Given an input automaton M and an alphabet partition Σ = Σr∪Σu, it checks whether θ(M) has the observer property. Σr denotes the set of relevant events, while Σu denotes the set of non-relevant events.

A restriction imposed on the input automaton M, for the purpose of applying the algorithm presented, is that it does not have cycles of non-relevant events. It is also assumed, without loss of generality, that the automaton M is trim.

The main algorithm is shown in Fig. 1. Its idea is to compute a nondeterministic automaton V = (Q, Σ, δ, {q0,V }), the so-called verifier, with states labelled by sets of up to two states of the input automaton M. The violation of the observer property is detected by the accessibility of a special state named Dead. The state set of the verifier is

\[ Q ⊆ \{x, y\} | x, y ∈ Q,M \} \cup \{\text{Dead}\}. \]

In addition to Dead, there are two types of states. States \{x, y\} with x ≠ y denote that M may reach the states x, y ∈ Q,M by traces with the same projection. In

\[ 1. \text{Input } M = (Q,M, Σ, δ,M, q0,M); \]

\[ 2. \text{Let } Q = \emptyset \text{ and } Q_T = \{\text{q0,M}\}; \]

\[ 3. \text{while } Q_T \neq Q \text{ do } \]

\[ 4. \text{Select } x ∈ Q_T \text{ - } Q; \]

\[ 5. \text{Let } Q_T = Q_T \text{ - } \{x\} \text{ and } Q = Q \cup \{x\}; \]

\[ 6. \delta(x); \]

\[ 7. \text{end} \]

\[ 8. \text{Return } V = (Q, Σ, δ, \{q0,V \}); \]
subroutine Delta(X)
9 Assume X = {x, y} (where possibly x = y);
10 for each σ ∈ En(X) = En M(x) ∪ En M(y) do
11 if σ ∈ Σ, then
12 if δ M(x, σ) and δ M(y, σ) then
13 δ(x, y, σ) = {δ M(x, σ), δ M(y, σ)};
14 Q T = Q T ∪ {δ M(x, σ), δ M(y, σ)};
15 else if δ M(x, σ) and En M(y, y) ∪ Σ ∅ = ∅ then
16 δ(x, y, σ) = {Dead};
17 Q = Q ∪ {Dead};
18 end
19 else
20 if δ M(x, σ) then
21 δ(x, y, σ) = δ M(x, σ) ∪ {δ M(x, σ), δ M(y, σ)};
22 Q T = Q T ∪ {δ M(x, σ), δ M(y, σ)};
23 end
24 if δ M(y, σ) then
25 δ(x, y, σ) = δ M(y, σ) ∪ {δ M(x, σ), δ M(y, σ)};
26 Q T = Q T ∪ {δ M(x, σ), δ M(y, σ)};
27 end
28 end
29 end

Fig. 2. Transition function subroutine Delta(X).

addition, the case x = y is possible in (2) and produces states of the form \(x\).

The verifier’s nondeterministic transition function δ: Q × Σ → 2Q is constructed according to the procedure Delta(X) in Fig. 2. At first, the only known state is the initial state \(q_0^M\). The other states are enumerated as they are reached from the initial state. The subroutine Delta(X) is called for each unvisited state in Q T that is not yet in Q, generating new states and transitions. Like the verifier presented in (Yoo and Lafortune, 2002), the OP-Verifier has its transition function defined differently for relevant and non-relevant events.

3.2 Properties

The main correctness result of the OP-Verifier algorithm is established in (Pena et al., 2008). The verifier contains Dead as an accessible state if and only if the observer property is violated.

Definition 2. Let \(V = (Q, Σ, Δ, q_0^M)\) be a verifier, and let \(X, Y \in Q\). The state Y is called accessible from X, if there exists a trace \(s \in Σ^*\) such that \(Y \in Δ(X, s)\). The state Y is called accessible in \(V\) if it is accessible from \(q_0^M\).

Theorem 1. (Pena et al., 2008) Let \(V\) be the verifier for automaton \(M\), let \(Σ \subseteq Σ\), and let \(θ: Σ^* → Σ^*\) be the natural projection. \(θ(M)\) satisfies the observer property if and only if state Dead is not accessible in \(V\).

The OP-Search algorithm considered in Section 4 below seeks to modify an unsuccessful verifier in such a way that the observer property becomes true. For this purpose, it needs to distinguish safe verifier states, where the observer property has been established, from other unsafe states. This distinction can be made using the accessibility of the special state Dead. Therefore, the state Q of the verifier is partitioned into two subsets:

\[ Q_V^{safe} = \{ X ∈ Q | Dead \text{ is accessible from } X \text{ in } V \} ; \]
\[ Q_V^{unsafe} = \{ X ∈ Q | Dead \text{ is not accessible from } X \text{ in } V \} . \]

States in \(Q_V^{unsafe}\) are considered as safe states, while states in \(Q_V^{safe}\) are unsafe. The following lemmas show some useful properties about the accessibility of verifier states.

Lemma 2. Let \(V = (Q, Σ, δ, (q_0^M))\) be a verifier for \(M = (Q^M, Σ, δ^M, q_0^M)\), and let \(a, b \in Q^M\).

(i) If \(b\) is accessible from \(a\) in \(M\), then \(b\) is accessible from \(\{a\}\) in \(V\).

(ii) If \(a\) is accessible in \(M\), then \(\{a\}\) is accessible in \(V\).

Proof. Consider a single transition between two states \(x, y \in Q^M\) on the path from \(a\) to \(b\), i.e., let \(σ \in Σ\) such that \(δ^M(x, σ) = y\). If \(σ \in Σ\), it follows by construction (line 13 in Fig. 2) that \(δ(x, σ) = \{y\}\). If \(σ \in Σ\), it follows from lines 21 and 25 that \(\{x, y\} \in δ(x, σ)\) and \(\{y\} \in δ(x, σ)\). Then the claim (i) follows by induction, and the claim (ii) follows because accessibility in \(V\) means accessibility from the initial state \(\{q_0^M\}\).

Lemma 3. Let \(V\) be the verifier for \(M = (Q^M, Σ, δ^M, q_0^M)\). Any transition to the state Dead originates in a state \(\{x, y\}\), where \(x, y \in Q^M\) and \(\{x, y\} = 2\).

Proof. The only way how a transition to Dead can be generated is by line 16 in Fig. 2. Because of the tests in line 12 and 15, this line can only be executed if \(δ(x, σ)\) and \(δ(y, σ)\) are not both defined, but at least one of them is. This means that \(x ≠ y\), which implies the claim.

Lemma 4. Let \(V\) be the verifier for \(M = (Q^M, Σ, δ^M, q_0^M)\). If Dead is accessible in \(V\), then there exists a state \(a \in Q^M\), an unobservable event \(η \in Σ\), and a trace \(t \in Σ^*\) such that \(Dead ∈ δ(\{a\}, η)^t\), and for all proper prefixes \(t' < t\) it holds that \(δ(\{a\}, η)^t') = 2\).

Proof. Since Dead is accessible in \(V\), there exists a path from the initial state \(\{q_0^M\}\) to Dead. Since Dead is not an initial state, this path is nonempty and, by Lemma 3, the last state entered before Dead is of the form \(\{x, y\}\) with \(\{x, y\} = 2\). Also, there exists a state \(a \in Q^M\) such that \(\{a\}\) appears on this path, for example \(q_0^M\). The subpath starting from the last of these states \(\{a\}\) satisfies the claim.

Lemma 4 shows how a failure of the observer property can be traced back to a non-relevant transition of the input automaton. This gives the starting point for amending the input automaton to satisfy the observer property in Section 4.

3.3 Complexity

The state set \(Q\) of the verifier \(V = (Q, Σ, Δ, q_0^M)\) for input automaton \(M = (Q^M, Σ, δ^M, q_0^M)\) contains unordered pairs of states from \(Q^M\) and possibly the state Dead. The number of states is thus bounded by

\[ |Q| ≤ \frac{1}{2}|Q^M|(|Q^M| + 1) + 1 . \]

For each transition \(δ^M(x, σ) = y\) in \(M\) there is exactly one transition originating from state \(\{x\}\) in \(V\), plus at most one transition from each state \(\{x, x'\}\) with \(x' ≠ x\). The state Dead has no outgoing transitions, so the following is an upper bound for the number of transitions in \(δ\),

\[ |δ| ≤ |Q^M||δ^M| . \]
1. Input automaton $M = (Q_M, \Sigma, \delta_M, q_0^M)$ and initial relevant event set $\Sigma_r \subseteq \Sigma$;
2. Construct verifier $V = (Q, \Sigma, \delta_V, q_0^V)$ for $M$ and $\Sigma_r$;
3. While $\text{Dead}$ is accessible in $V$
do{
  4. Find $a \in Q_M^*, \eta \in \Sigma_u$, and $t \in \Sigma^+$ such that $\delta(a, \eta) = \text{Dead}$ and for all $\eta' < t$, $|\delta(a, \eta')| = 2$;
  5. Choose new event $\eta' \notin \Sigma_r$;
  6. Let $\Sigma_r = \Sigma_r \cup \{\eta'\}$ and $\Sigma = \Sigma \cup \{\eta'\}$;
  7. Define $\hat{M} = (Q_M, \Sigma, \delta_M, q_0^M)$ where $\delta^\hat{M}$ is the same as $\delta_M$ except $\delta^M(a, \eta)$ is undefined and $\delta^\hat{M}(a, \eta') = \delta^M(a, \eta)$;
  8. Construct new verifier $V$ for $\hat{M}$ and $\hat{\Sigma}_r$;
  9. Let $V = V, M = M$, and $\Sigma_r = \Sigma_r$;
10. end
11. Return $\hat{M} = M$ and $\hat{\Sigma}_r = \Sigma_r$;

Fig. 3. The OP-Search Algorithm.

Here, $|\delta|$ denotes the cardinality of the transition relation induced by the transition function $\delta$.

Noting that $|Q_M|^3 \leq |\delta_M| + 1$ for accessible $M$ and $|\delta_M| \leq |Q_M||\Sigma|$ for deterministic $M$, the size of the verifier is bounded by

\[
O(|Q_M|^3|\delta_M|) = O(|Q_M|^3|\Sigma|).
\]

The transition function algorithm $\Delta(X)$ can be completed by visiting every transition in $\delta$ a single time, so (5) also gives a worst-case bound for the time complexity of the OP-Verifier algorithm. This is an improvement over methods based on observation equivalence, which run in $O(|Q_M|^3)$ time (Bolognesi and Smolka, 1987).

4. COMPUTATION OF OBSERVERS

This section presents the OP-Search algorithm, which operates on a verifier and modifies its relevant event set, until a projection satisfying the observer property is found.

After constructing the verifier for the input automaton, the OP-Search algorithm identifies the causes that lead to failure of the observer property, i.e., the paths leading to the verifier’s Dead state. If present, one of the paths to the verifier’s Dead state is chosen, which by the properties of the verifier is guaranteed to contain a transition labelled with a non-relevant event. By choosing such a transition and making its event relevant, the chosen path to the Dead state is removed. The procedure is repeated until all paths to Dead are eliminated, resulting in a modified automaton and relevant event set satisfying the observer property.

4.1 The OP-Search Algorithm

The OP-Search algorithm is shown in Fig. 3. On input of automaton $M$ and relevant event set $\Sigma_r$, the first step is to construct the verifier $V$ for $M$ and check whether the observer property is satisfied. According to Theorem 1, this is the case if the state Dead is not accessible in $V$, in which case the algorithm terminates and returns the unmodified automaton $M$.

Otherwise, if Dead is accessible, the automaton and event set need to be modified. In line 4, the algorithm selects a path from some state $\{a\}$ to Dead that starts with a non-relevant event $\eta$ and passes only through states $\{x_i, y_i\}$, $i = 1, \ldots, n$, such that $x_i \neq y_i$. Such a path exists according to Lemma 4. The path’s initial $\eta$-transition is relabelled using a new relevant event $\eta'$ in lines 5–7. Then a new verifier is constructed for the modified automaton, and the procedure starts over.

Clearly, the algorithm only terminates when a verifier without an accessible state Dead has been obtained, so the resultant automaton and projection satisfy the observer property. While the choice of events $\eta'$ replaced in line 4 is simple and straightforward to implement, and gives reasonable results, it is not necessarily optimal.

Also, the input automaton $M$ is modified to $\hat{M}$ replacing some non-relevant events $\eta$ on transitions by arbitrary new events $\eta'$. Yet, this relabelling does not affect how the automaton may synchronise with other components. For compositional nonblocking verification (Pena et al., 2009; Flordal and Malik, 2009), the replaced events $\eta$ and the introduced events $\eta'$ are not shared with any other automaton in the remainder of the system being analysed, so this relabelling does not affect whether the complete system is nonblocking or not. Therefore, the resultant automaton $\hat{M}$ can be used as an abstraction in nonblocking verification. Likewise, the relabelling can be used for purposes of modular synthesis (Feng and Wonham, 2006; Hill and Tilbury, 2006), because only deterministic automata are constructed, so it is always possible to relate any events disabled later in the synthesis to the corresponding transitions in the input automaton.

The result of OP-Search, the projection $\hat{\eta}(\hat{M})$, is used as a reporter map (Wong and Wonham, 2004), which relabels some non-relevant transitions using the new events $\eta'$ and erases the others.

4.2 Analysis

The correctness of the result of the OP-Search algorithm is easily confirmed by the fact that, at termination, a verifier with the state Dead not accessible exists for the result automaton. Theorem 5 follows directly from Theorem 1.

Theorem 5. Let $\hat{M}$ and $\hat{\Sigma}_r$ be the output of OP-Search, and let $\hat{\eta}: \Sigma^* \rightarrow \hat{\Sigma}_r$ be the resultant natural projection. Then $\hat{\eta}(\hat{M})$ has the observer property.

To see that the OP-Search algorithm terminates, it is enough to note that each iteration of the loop selects a non-relevant transition from the input automaton $M$ and relabels it using a relevant event. Since there is only a finite number of transitions in $M$, the loop must terminate. In the following, a more detailed analysis that leads to a tighter bound on the number of iterations is carried out.

Firstly, if Dead is not reachable from a verifier state $\{b\}$, then this state will retain this property after execution of the main loop of OP-Search. That is, once a state is recognised as safe, it remains safe for the rest of the algorithm.

Theorem 6. Let $V$ be a verifier for $M = (Q_M, \Sigma, \delta_M, q_0^M)$ and $\Sigma_r \subseteq \Sigma$, and let $V$ be constructed from $V$ according to lines 4–9 of the OP-Search algorithm. Furthermore, let $b \in Q_M$. Then $\{b\} \subseteq Q_V^{\text{rec}}$ implies $\{b\} \subseteq Q_V^{\text{rec}}$. 

419
Proof. Let $b \in Q^M$ such that $\{b\} \in Q_{V}^{\text{gcc}}$. That is, there exists a trace $s \in \Sigma^*$ such that $\text{Dead} \in \hat{\delta}(\{b\}, s)$. (6)

Let $s = \sigma_1 \ldots \sigma_n$ and $X_0, \ldots, X_n \in Q$ such that $X_0 = \{b\}$, $X_{i+1} \in \hat{\delta}(X_i, \sigma_i)$ for $i = 0, \ldots, n - 1$, and $X_n = \text{Dead}$. Furthermore let $m \leq n$ be the smallest index such that $X_m = \text{Dead}$ or $a \in X_m$, where $a \in Q^M$ with $\{a\} \in Q_{V}^{\text{gcc}}$ is the state chosen in line 4.

Consider an arbitrary index $i < m$. Then $\sigma_i \neq \eta$, where $\eta \in \Sigma$, is the new event introduced in line 6, because $\eta$ is only enabled in state $a$ of $M$, and $a \notin X_i$. It follows by construction that $X_{i+1} \in \hat{\delta}(X_i, \sigma_i)$, and by induction that $X_m$ is accessible from $\{b\}$ in $V$.

If $X_m = \text{Dead}$, then clearly $\{b\} \in Q_{V}^{\text{gcc}}$. Otherwise $a \in X_m$ such that $X_m$ is accessible from $\{b\}$ in $V$. This implies by construction of $V$ that $a$ is accessible from $b$ in $M$. Then $\{a\}$ also is accessible from $\{b\}$ in $V$ by Lemma 2, and since $\{a\} \in Q_{V}^{\text{gcc}}$, this implies $\{b\} \in Q_{V}^{\text{gcc}}$.

Theorem 7. The execution of OP-Search with input $M = (Q^M, \Sigma, \delta^M, \theta^M)$ terminates in at most $|Q^M||\Sigma_u|$ iterations of the main loop, where $Q_{V}^{\text{gcc}} = \{b \in Q^M | \{b\} \in Q_{V}^{\text{gcc}} \}$. (7)

Proof. Given verifier $V = (Q, \Sigma, \delta, \{q_0^M\})$, consider the set $T_V = \{(b, \sigma) \in Q^M \times \Sigma_u | \{b\} \in Q_{V}^{\text{gcc}} \land \exists X \in \delta(b, \sigma) : |X| = 2 \}$. (8)

It is enough to show that $|T_V| < |T_V|$ after each iteration of the loop of OP-search.

In line 4, the algorithm chooses $\{a\} \in Q_{V}^{\text{gcc}}$ and $\eta \in \Sigma_u$ such that $(a, \eta) \in T_V$. Then $\delta(a, \eta) = \{a, x\}$ for some $x \neq a$ and $\delta^M(a, \eta) = x$ by construction of $V$. In line 7, the algorithm constructs $M = (Q^M, \Sigma, \delta^M, \theta^M)$ such that $\delta^M(a, \eta)$ is undefined and $\delta^M(a, \eta') = x$. Let $V = (Q, \Sigma, \hat{\delta}, \{q_0^M\})$. It will be shown that $T_V \subseteq T_V - \{(a, \eta)\}$.

Clearly, $(a, \eta) \notin T_V$, because $\delta^M(a, \eta)$ is undefined by construction in line 7. Consider $(b, \sigma) \in T_V$. Then $\{b\} \in Q_{V}^{\text{gcc}}$, and it follows from Theorem 6 that $\{b\} \in Q_{V}^{\text{gcc}}$. Furthermore, since $(b, \sigma) \in T_V$, it holds that $\sigma \in \Sigma_u$, and there exists $X \in \delta b, \sigma)$ such that $|X| = 2$. By construction of $\delta^M$, and since $\eta \in \Sigma_u$, it follows that $\delta^M(b, \sigma) \subseteq \delta^M(b, \sigma)$, because the only transition added to $\delta^M$ is in $\delta^M(a, \eta')$ with $\eta' \notin \Sigma_u$. Then $X \in \delta b, \sigma)$ with $|X| = 2$, i.e., $(b, \sigma) \in T_V$.

Therefore, $T_V \subseteq T_V - \{(a, \eta)\}$ and $|T_V| \leq |T_V| - 1$. 4.3 Complexity

By Theorem 7, the number of iterations of the main loop in OP-Search is bounded by $|Q_{V}^{\text{gcc}}||\Sigma_u|$. Each iteration involves a search of the verifier to determine whether the state $\text{Dead}$ is accessible, a modification of the automaton $M$, and the construction of a new verifier. The complexity of these steps is determined by the number of transitions of the verifier, which is bounded by $O(|Q^M||\delta^M|)$ according to (4). This results in an worst-case time complexity for the OP-Search algorithm of

$$O(|Q^M||\delta^M||Q_{V}^{\text{gcc}}||\Sigma_u|) = O(|Q^M||\Sigma|_new^2).$$ (9)

In comparison, the method of (Wong and Wonham, 2004), which computes an optimal reporter map, runs in $O(|Q^M||\delta^M|) = O(|Q^M||\Sigma|_new^2)$, and the algorithm in (Feng and Wonham, 2010), which determines a suitable relevant event set only by choosing events from the given alphabet, has a worst-case time complexity of $O(|Q^M||\delta^M|_new^2) = O(|Q^M||\Sigma|_new^2)$.

5. EXAMPLE

To illustrate the OP-search algorithm, a pair $(M, \Sigma_r)$ with a verifier that reaches the state $\text{Dead}$ is presented. After a few iterations of OP-search, the resulting pair $(M, \Sigma_r)$ is such that $\text{Dead}$ is not accessible in $V_M$.

Let $M = (Q^M, \Sigma, \delta^M, \theta^M)$ be the automaton presented in Figure 4. The set of relevant events is $\Sigma_r = \{a, b, \tau\}$. The OP-verifier $V_M$ that results from the execution of the OP-search algorithm is shown in Fig. 5.

Fig. 4. Automaton $M$.  

Fig. 5. Verifier $V_M$ obtained for input $(M, \Sigma_r)$.

State $\text{Dead}$ is accessible in $V_M$, so we can conclude that $\theta(M)$ is not an OP-abstraction. The algorithm OP-search is applied to the pair $(M, \Sigma_r)$.

The loop of algorithm OP-search (line 4 to 7) is executed until state $\text{Dead}$ is not accessible in the verifier. Let state $5 \in Q^M$ and event $z$ be chosen. This pair passes the test of line 4. The algorithm modifies $M$, by relabelling the transition $\delta^M(5, z) = 6$ to $\delta^M(5, z_1) = 6$. The result, $\hat{M}$, is shown in Fig. 6.

Fig. 6. Automaton $\hat{M}$, resulting from one iteration of OP-search.

The verifier that results from the application of the OP-verifier to the pair $(\hat{M}, \Sigma_r)$, where $\Sigma_r = \{a, b, z_1, \tau\}$, is presented in Fig. 7.

Note that $\{2\}, \{4\}, \{6\}, \{7\}$ are safe states in the verifier of Fig. 5, and $\{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$ are safe in the verifier of Fig. 7—as expected by Theorem 6.
Two more iterations occur before OP-search converges, relabelling the transition \( \delta^M(1, x) = 4 \) to \( \delta^M(1, x_1) = 4 \) and the transition \( \delta^M(0, z) = 2 \) to \( \delta^M(0, z_2) = 2 \). The new events \( x_1 \) and \( z_2 \) are included in the relevant event set. The final automaton \( \hat{M} \) and the corresponding OP-verifier \( V_{\hat{M}} \) are presented in Fig. 8.

![Fig. 7. Verifier \( V_{\hat{M}} \) obtained given input (\( \hat{M}, \hat{\Sigma}_r \)).](image)

Since state \textit{Dead} is not accessible in the verifier \( V_{\hat{M}} \) obtained from the pair \((M, \Sigma_r)\), the natural projection \( \hat{\theta}(\hat{M}) \) has the observer property, for \( \hat{\theta} : \hat{\Sigma}^* \rightarrow \Sigma^*_r \). The only non-relevant event that remains in the result is \( x \) on the transition from 3 to 5 in \( \hat{M} \). This is the best possible solution for this example, although this optimality is not guaranteed in general.

6. CONCLUSIONS

The OP-Search proposed in this paper is an efficient way to compute observers for DES models. The modification of the set of events in the original model, made by relabelling transitions, allows the OP-abstraction to be obtained by natural projections instead of by reporter maps. On the other hand, the selection of transitions to be relabelled is done in such a way that the resulting modified model remains suitable for the control problem where it is to be considered. The authors are currently investigating how the OP-Search algorithm can be improved to lead to minimal OP-abstractions.

REFERENCES


