On Specification Transparency: Towards A Formal Framework for Designer Comprehensibility of Discrete-Event Control Specifications in Finite Automata

Manh Tung Pham, Amrith Dhananjayan and Kiam Tian Seow

***Abstract*— In control of discrete-event systems (DES's), specifying control requirements in automata is not a trivial task. For many DES applications, designers are often confronted with the long-standing problem of uncertainty in specification, namely: how do we know that a specification automaton does indeed model the intended control requirement? Towards a formal framework that helps mitigate this uncertainty for designer comprehensibility, in this paper, we introduce and develop a new specification concept of automaton transparency, and investigate the problem of maximizing the transparency of specification automata for DES's. In a transparent specification automaton, events that are irrelevant to the specification but can occur in the system are 'hidden' in self-loops. Different automata of the same specification on a DES can be associated with different sets of such irrelevant events; and any such automaton is said to be the most transparent if it has an irrelevant event set of maximal cardinality. The transparency maximization problem is theoretically formulated and a provably correct solution algorithm is obtained. Given a specification automaton for a DES, the transparent specification automaton produced by the algorithm is a more comprehensible structure, essentially showing the precedence ordering among events from a minimal cardinality set that is relevant in modeling some requirement for the DES, and should aid designers in clarifying if the requirement prescribed is the one intended.**

***Index Terms*— Discrete event systems, specification automata, language relevance, transparency.**

I. INTRODUCTION

Supervisory control of discrete-event systems (DES's) presents a formal and effective framework [2]–[4] to model and control complex systems. In this framework, a system to be controlled and a control specification are often modeled as finite-state automata [5], following which a supervisor can be automatically synthesized to control the system in conformance to the specification.

In practice, an automaton is often manually prescribed by a system designer following a linguistic description (verbal or textual) of some control requirement; or it may be automatically translated from a requirement already expressed as some temporal logic specification [6]. Deciding if a specification automaton actually reflects the intended control requirement correctly and completely lacks formal theoretical support, and is a challenging task, especially for large DES's. The uncertainty of whether or not an intended requirement is correctly modeled by an automaton has often been encountered in many applications of the automata-based DES framework (e.g., in robotics [7], automated manufacturing [8]–[10], and intelligent service transportation [11]).

In this paper, such uncertainty is resolved as the problem of maximizing the transparency of control specifications prescribed in finite automata. In what we call a transparent specification automaton, events that are irrelevant to the specification but can occur in the system are 'hidden' in self-loops; while events that are relevant to the specification are highlighted in diligent transitions (i.e., those connecting distinctly different states). Different automata of the same specification on a DES can be associated with different sets of relevant events. The most (or maximally) transparent specification automaton essentially shows the precedence ordering among events from a minimal cardinality set that is relevant to the requirement. Conversely, it hides events from the irrelevant event set of maximal cardinality.

A preliminary conference version of this paper appeared in [1].

This research is funded by the Singapore Ministry of Education, under NTU-AcRF Tier 1 Grant No: RG65/07.

The authors are with the Division of Computing Systems, School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. {Pham0028,Amri0005,asktseow}@ntu.edu.sg

Such transparency could more readily highlight the linguistic expression of the specification; and should help towards resolving the long-standing problem in specification, namely: how do we know that a specification in automata does indeed capture the intended control requirement? For an intuitive example, the reader might want to skip ahead to Section VI for a maximally transparent specification automaton [see Fig. 1(d)] of a first come, first served control requirement for a resource allocation system.

Our work falls within the research scope of system designer comprehensibility, which is a current major concern in industrial applications of supervisory control [12]. Unlike in this paper which seeks to provide support for understanding design specifications (i.e., on "what to control"), past and existing research seeks to provide support for achieving supervisor clarity, and at the control-action level (i.e., on "how to control"). For example, a temporal logic framework is proposed in [13] to compute individual controls on (controllable) events as readable temporal logic formulas for the temporal-safety class of specifications. In [14], techniques are developed to generate and attach propositional formulae called guards to a given supervisor automaton, which are also believed to comprehensibly model the logical conditions under which individual events are enabled or disabled. In contrast, the transparency maximization problem proposed and investigated in this paper is to recast a specification in automata into a more comprehensible structure. The problem solution can be used to support the control design framework [2], by assisting designers at the outset to ascertain the correctness of specification automata prior to supervisor synthesis. As a good engineering practice, it is important for designers to understand the formal specifications prescribed and assess their correctness first. Once this is done, supervisor comprehensibility, in our opinion, may be optional in some DES applications as long as provably correct algorithms (e.g., [2]) are applied in supervisor synthesis.

Technically related research includes works (e.g., [15], [16]) that focus on minimizing or reducing the number of states in a supervisor automaton to achieve economy of implementation. The procedures developed might lead to transparent automata in certain cases. However, our problem is different as it focuses on maximizing transparency of specification automata. In so doing, we attempt to render a specification automaton more understandable for a system designer, as opposed to state reduction in a supervisor. Computing a maximally transparent specification automaton may, as a byproduct, minimize or reduce the number of states in it.

The rest of this paper is organized as follows. In Section II, we review preliminary concepts in languages and automata theory that are most relevant to this paper. We then define the concepts of a transparent automaton and a relevant specification language (Section III-A), and formally state the problem of finding a maximally transparent specification automaton (Section III-B). In Section IV, we provide the detailed problem analysis. Our first main result (Theorem 1) establishes the connection between the two defined concepts, motivating the development of a formal language relevance verification procedure (Section V-A, Theorem 2) and a procedure to compute a set of relevant events of minimal cardinality for a given specification language (Section V-B). Based on the two developed procedures, a provably correct solution algorithm (Algorithm 1, Theorem 3) for the problem of finding a maximally transparent specification automaton is then presented in Section V-C. In Section VI, illustrative examples are provided to demonstrate the concept of a transparent specification synthesized using Algorithm 1. Finally, Section VII concludes the paper and points to some future work.

II. Preliminaries: Languages and Automata

Let Σ be a finite alphabet of symbols representing individual events. A string is a finite sequence of events from Σ. Denote Σ^* as the set of all strings from Σ including the empty string ε. A string s' is a prefix of s if $(\exists t \in \Sigma^*)\ s't = s$, where $s't$ is the string obtained by catenating t to s'.

A language L over Σ is a subset of Σ^*. Say L_1 is a sublanguage of L_2 if $L_1 \subseteq L_2$. The prefix closure $\bar{L}$ of a language L is the language consisting of all prefixes of its strings. Clearly $L \subseteq \bar{L}$, because any string s in Σ^* is a prefix of itself. A language L is prefixed-closed if $L = \bar{L}$.

Given $\Sigma^1 \subseteq \Sigma^2$, the natural projection $P_{\Sigma^2,\Sigma^1} : (\Sigma^2)^* \to (\Sigma^1)^*$, which erases from a string $s \in (\Sigma^2)^*$ every event $\sigma \in (\Sigma^2 - \Sigma^1)$, is defined recursively as follows: $P_{\Sigma^2,\Sigma^1}(\varepsilon) = \varepsilon$, and $(\forall s \in$

$(\Sigma^2)^*)(\forall\sigma \in \Sigma^2)$, $P_{\Sigma^2,\Sigma^1}(s\sigma) = P_{\Sigma^2,\Sigma^1}(s)\sigma$ if $\sigma \in \Sigma^1$, and $P_{\Sigma^2,\Sigma^1}(s\sigma) = P_{\Sigma^2,\Sigma^1}(s)$, otherwise.

For $L \subseteq (\Sigma^2)^*$, $P_{\Sigma^2,\Sigma^1}(L) \subseteq (\Sigma^1)^*$ denotes the language $\{P_{\Sigma^2,\Sigma^1}(s) \mid s \in L\}$.

If a language is regular [5], then it can be generated by an automaton. An automaton G is a 5-tuple $(Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite set of states, Σ is the finite set of events, $\delta : \Sigma \times Q \to Q$ is the (partial) transition function, q_0 is the initial state and $Q_m \subseteq Q$ is the subset of marker states.

In this paper, a language is assumed to be regular.

Write $\delta(\sigma, q)!$ to denote that $\delta(\sigma, q)$ is defined, and $\neg\delta(\sigma, q)!$ to denote that $\delta(\sigma, q)$ is not defined. The definition of δ can be extended to $(\Sigma)^* \times Q$ as follows: $\delta(\varepsilon, q) = q$, and $(\forall\sigma \in \Sigma)(\forall s \in (\Sigma)^*)\delta(s\sigma, q) = \delta(\sigma, \delta(s, q))$.

The behaviors of automaton G can then be described by the prefix-closed language $L(G)$ and the marked language $L_m(G)$. Formally, $L(G) = \{s \in (\Sigma)^* \mid \delta(s, q_0)!\}$ and $L_m(G) = \{s \in L(G) \mid \delta(s, q_0) \in Q_m\}$.

A state $q \in Q$ is reachable if $(\exists s \in (\Sigma)^*)$ $\delta(s, q_0) = q$, and coreachable if $(\exists s \in (\Sigma)^*)$ $\delta(s, q) \in Q_m$. Automaton G is reachable if all its states are reachable, and coreachable if all its states are coreachable and so $\overline{L_m(G)} = L(G)$. G is then said to be trim if it is both reachable and coreachable. If G is not reachable, then a reachable automaton, denoted by $Ac(G)$, can be computed by deleting from G every state that is not reachable. Thus, $Ac(G)$ generates the same prefix-closed and marked languages as G. If G is not trim, then a trim automaton, denoted by $Trim(G)$, can be computed by deleting from G every state that is either not reachable or not coreachable. Therefore, $Trim(G)$ has no unreachable states and no uncoreachable states, and generates the same marked language as G.

III. Problem Concepts and Description

A. Automaton Transparency and Language Relevance

Definition 1: Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, and a language L such that $L = L_m(A)$, where automaton $A = (X, E, \xi, x_0, X_m)$. If A is said to be a specification automaton (of L for DES G), then 1) $E = \Sigma$, 2) $\overline{L_m(A) \cap L_m(G)} = L(A) \cap L(G)$, and 3) A is trim.

Intuitively, a well-defined specification automaton for DES G models a task (marked) sublanguage of G over event set Σ. The sublanguage $L_m(A) \cap L_m(G)$ is well modeled in that every common prefix string in $L(A) \cap L(G)$ can be extended to a marked string in $L_m(A) \cap L_m(G)$, thereby specifying an uninhibited sequence of event executions to complete some task[1].

Definition 2: A specification automaton A (for DES G) is said to be Σ_{irr}-transparent if $\Sigma_{irr} \subseteq \Sigma$ is a set of strictly self-loop events in A, i.e., $(\forall\sigma \in \Sigma_{irr})(\forall x \in X)(\xi(\sigma, x)! \Rightarrow \xi(\sigma, x) = x)$.

A Σ_{irr}-transparent specification automaton A has all the events in $\Sigma_{irr} \subseteq \Sigma$ 'hidden' in self-loops, thus showing more explicitly the precedence ordering of the rest of the events deemed relevant to the intended requirement that it specifies. In other words, those events in Σ_{irr} can be considered irrelevant to the specification, although they can occur in the DES G. We postulate that for the most (or maximally) transparent automaton A, the irrelevant event set Σ_{irr} must be of maximal cardinality.

Definition 3: A language $K \subseteq L_m(G)$ is said to be Σ_{rel}-relevant with respect to (w.r.t) G if $(\forall s, s' \in (\Sigma)^*)$ for which $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s')$, the following two conditions are satisfied:

1) $(\forall\sigma \in \Sigma)[(s\sigma \in \overline{K}$ and $s' \in \overline{K}$ and $s'\sigma \in L(G)) \Rightarrow s'\sigma \in \overline{K}]$.
2) $[s \in K$ and $s' \in \overline{K} \cap L_m(G)] \Rightarrow s' \in K$.

Informally, Condition 1 asserts that the projected language of $\overline{K}$ onto events from Σ_{rel} is sufficient to highlight the relevant precedence ordering of events as specified. Condition 2 asserts that the projected language of K can sufficiently highlight the relevant marking as specified for G. Thus, when a language $K \subseteq L_m(G)$ is Σ_{rel}-relevant w.r.t G, it means that the precedence order among events from Σ_{rel} contains the essence of the specification for G that K embodies. Σ_{rel} is called a relevant event set of such a K.

Remark 1: Note that language relevance w.r.t a set of relevant events and language observability [17] w.r.t a set of observable events may share identical mathematical conditions, but their concepts

[1]In this work, following the standard treatment in supervisory control theory, we consider a DES G as given. In practice, G is often constructed by system designers through an iterative process of modeling and re-modeling. How to construct G to correctly and completely model a system of interest is an open design problem that is beyond the scope of this paper.

are fundamentally different: events in a relevant event set need not be observable in the control-theoretic sense, but are identified as a collective set that can prescribe the essence of a specification in an automaton.

B. Problem Statement

We now formally state the problem of finding a maximally transparent specification automaton A that models a given language $K \subseteq L_m(G)$ on DES G, i.e., $L_m(A) \cap L_m(G) = K$.

Problem 1: Given DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ and a specification language $K \subseteq L_m(G)$, construct a specification automaton A (according to Definition 1) so that:

1) A is Σ_{irr}-transparent and $L_m(A) \cap L_m(G) = K$;
2) $(\forall \Sigma' \subseteq \Sigma, |\Sigma'| > |\Sigma_{irr}|)$, there is no Σ'-transparent specification automaton A' such that $L_m(A') \cap L_m(G) = K$.

For the language K under DES G, Condition 1 specifies the Σ_{irr}-transparency of A and Condition 2 specifies the maximal cardinality of the irrelevant event set $\Sigma_{irr} \subseteq \Sigma$ associated with A.

IV. Problem Analysis

In what follows, if a language $K \subseteq L_m(G)$ is Σ_{rel}-relevant, then a specification automaton A that is $(\Sigma - \Sigma_{rel})$-transparent can be synthesized such that $L_m(A) \cap L_m(G) = K$. This is formally stated in Theorem 1. The proof of this fundamental result requires a procedure called $Trans$.

Procedure $Trans$ computes and returns a E_{irr}-transparent automaton A from a given automaton H and an event subset E_{irr}. For $E_{rel} = E - E_{irr}$, Step 1 and Step 2 of $Trans$ involve computing an automaton A'' that is due to the projection of the languages of H onto E_{rel}^*, i.e., $L_m(A'') = P_{E,E_{rel}}(L_m(H))$ and $L(A'') = P_{E,E_{rel}}(L(H))$; and Step 3 adds additional self-loop transitions of events in E_{irr} to A'' to obtain the resulting automaton A. As a result, the procedure has exponential time complexity of $O(2^{|Y|})$, where $|Y|$ is the state size of the input automaton H. This exponential time complexity, however, can be avoided if H has some special structure w.r.t E_{rel}, which will be discussed later in Section V-C.

The following lemma summarizes important properties of the computed automaton A.

Procedure $Trans(H, E_{irr})$

```
Input: Automaton $H = (Y, E, \zeta, y_0, Y_m)$ and an event subset
       $E_{irr} \subseteq E$;
Output: An automaton $A = (X, E, \xi, x_0, X_m)$ that is
        $E_{irr}$-transparent;
begin
    Let $\pi : X' \to 2^Y - \{\emptyset\}$ be a bijective mapping and
    $E_{rel} = E - E_{irr}$;
    Step 1: Compute $A' = (X', E_{rel}, \xi', x'_0, X'_m)$:
        • $x'_0 \in X'$ with
          $\pi(x'_0) = \{\zeta(s, y_0) \mid P_{E,E_{rel}}(s) = \varepsilon\}$;
        • $X'_m = \{x' \in X' \mid (\exists s \in L_m(H))\zeta(s, y_0) \in \pi(x')\}$;
        • $(\forall \sigma \in E_{rel})(\forall x' \in X')$ $(\xi'(\sigma, x')!$ if and only if
          $(\exists s\sigma \in L(H))\zeta(s, y_0) \in \pi(x'))$;
          When defined, $\xi'(\sigma, x') = x''$ with
          $\pi(x'') = \{\zeta(s', y) \mid y \in \pi(x'), P_{E,E_{rel}}(s') = \sigma\}$;
    Step 2: Trim $A'$ to get $A'' = (X, E_{rel}, \xi, x_0, X_m)$:
    $A'' = Trim(A')$;
    Step 3: Compute $A$ from $A''$:
        • $(\forall \sigma \in E_{irr})(\forall x \in X)$ if $(\exists y \in \pi(x))\zeta(\sigma, y)!$ then
          add a self-loop transition for $\sigma$ at state $x$: $\xi(\sigma, x) = x$;
        • The resulting automaton is the output automaton
          $A = (X, E, \xi, x_0, X_m)$;
    Return $A$;
```

Lemma 1: Let $H = (Y, E, \zeta, y_0, Y_m)$, $E_{irr} \subseteq E$, $E_{rel} = E - E_{irr}$ and $A = Trans(H, E_{irr})$. Then:

1) A is E_{irr}-transparent.
2) $(\forall s \in E^*)(\forall \sigma \in E)[s\sigma \in L(A) \Rightarrow (\exists s' \in L(H))(s'\sigma \in L(H)$ and $P_{E,E_{rel}}(s') = P_{E,E_{rel}}(s))]$.
3) $(\forall s \in L_m(A))(\exists s' \in L_m(H))[P_{E,E_{rel}}(s') = P_{E,E_{rel}}(s)]$.
4) $L_m(A) \supseteq L_m(H)$ and $L(A) \supseteq L(H)$.

Proof: To begin with, let A'' be the automaton generated in Step 2 of $Trans$. It is clear that $L_m(A'') = P_{E,E_{rel}}(L_m(H))$ and $L(A'') = P_{E,E_{rel}}(L(H))$. Since A is constructed from A'' in Step 3 of $Trans$ by adding self-loop transitions for events in E_{irr}, it is also clear that $(\forall s \in L(A))$ $P_{E,E_{rel}}(s) \in L(A'')$ and $P_{E,E_{rel}}(s) \in L(A)$.

The statements of the lemma can now be proved as follows.

1) Since every event in E_{irr} is only added to the transition structure of A in Step 3 of $Trans$ as a strictly self-loop event, it is clear that A is E_{irr}-transparent.
2) Let $s \in E^*$ and $\sigma \in E$ such that $s\sigma \in L(A)$. Then, $P_{E,E_{rel}}(s\sigma) \in L(A'')$. Let $t = P_{E,E_{rel}}(s)$. We need to show that there exists $s' \in L(H)$ such that $s'\sigma \in L(H)$ and $P_{E,E_{rel}}(s') = t$, as follows.
    - If $\sigma \in E_{rel}$, $P_{E,E_{rel}}(s\sigma) = t\sigma \in L(A'')$.

Therefore, since $L(A'') = P_{E,E_{rel}}(L(H))$, there exists $t' \in L(H)$ such that $P_{E,E_{rel}}(t') = t\sigma$. Since $\sigma \in E_{rel}$, t' must end with σ, i.e., $t' = s'\sigma$ for some $s' \in L(H)$. In other words, $P_{E,E_{rel}}(s') = t$. Hence the statement.

- On the other hand, if $\sigma \in E_{irr}$ then, by Step 3 of $Trans$, there exits $s' \in L(H)$ such that $P_{E,E_{rel}}(s') = t$ and $s'\sigma \in L(H)$. Hence the statement.

3) Let $s \in L_m(A)$ and $t = P_{E,E_{rel}}(s)$. An argument similar to that in the proof of the previous statement leads to $t \in L_m(A'')$ and therefore, there exits $s' \in L_m(H)$ such that $P_{E,E_{rel}}(s') = t$, since $L_m(A'') = P_{E,E_{rel}}(L_m(H))$. Hence the statement.

4) - *Proof of* $L(A) \supseteq L(H)$:
   Assume that $s \in L(H)$. We need to show that $s \in L(A)$, as follows.
     - Since $s \in E^*$, $s = s_0t_0s_1t_1...s_nt_n$, where $(\forall 0 \leq i \leq n)\ s_i \in E^*_{rel}$ and $t_i \in E^*_{irr}$ for some integer $n \geq 0$.
     - Let $u = s_0s_1...s_n$, then $u \in L(A)$. Let $y_i = \xi(s_0s_1...s_i, x_0)$ be the state of A after the execution of string $s_0s_1...s_i$, $0 \leq i \leq n$.
     - Since $s_0 \in L(A)$, $t_0 \in E^*_{irr}$, by Step 3 of $Trans$, $\xi(t_0, y_0) = y_0$. Therefore $s_0t_0 \in L(A)$ and $\xi(s_0t_0, x_0) = y_0$. Similar arguments lead to $s_0t_0s_1t_1...s_it_i \in L(A)$ and $\xi(s_0t_0s_1t_1...s_it_i, x_0) = y_i$ for all $0 \leq i \leq n$. Therefore $s \in L(A)$.

   Hence $L(A) \supseteq L(H)$.
   - *Proof of* $L_m(A) \supseteq L_m(H)$:
   Assume that $s \in L_m(H)$. We need to show that $s \in L_m(A)$, as follows.
     - Since $s \in L_m(H)$, $s \in L(H)$, which implies $s \in L(A)$ since $L(H) \subseteq L(A)$.
     - Let $x = \xi(s, x_0)$ be the state of A after the execution of s. Let $t = P_{E,E_{rel}}(s)$. Since $s \in L_m(H)$, we have $t \in L_m(A'')$.
     - Furthermore, by Step 3 of $Trans$, we also have $\xi(t, x_0) = x$. Therefore x is a marker state in A, which implies that $s \in L_m(A)$.

   Hence $L_m(A) \supseteq L_m(H)$.

■

We may now state our first main result.

Theorem 1: Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, a language $K \subseteq L_m(G)$ and an event subset $\Sigma_{irr} \subseteq \Sigma$. There exists a specification automaton A that is Σ_{irr}-transparent for G such that $K = L_m(A) \cap L_m(G)$ if and only if K is $(\Sigma - \Sigma_{irr})$-relevant w.r.t G.

Proof: Let $\Sigma_{rel} = \Sigma - \Sigma_{irr}$. For economy of notation, let P denote the natural projection $P_{\Sigma,\Sigma_{rel}}$.

(If:) Assume K is Σ_{rel}-relevant w.r.t G. We present a constructive proof to show that a Σ_{irr}-transparent specification automaton A for DES G (according to Definitions 1 and 2) exists such that $K = L_m(A) \cap L_m(G)$.

Let H be a trim automaton such that $L(H) = \overline{K}$ and $L_m(H) = K$. We then construct the specification automaton A from H using $Trans$: $A = Trans(H, \Sigma_{irr})$.

By Lemma 1, A is Σ_{irr}-transparent. To show our construction works, we need to show that A can be a specification automaton of Definition 1 modeling K on G, i.e., $\overline{K} = L(A) \cap L(G)$ and $K = L_m(A) \cap L_m(G)$.

By Lemma 1, $K \subseteq L_m(A)$ and $\overline{K} \subseteq L(A)$. Therefore, since $K \subseteq L_m(G)$, $\overline{K} \subseteq L(A) \cap L(G)$ and $K \subseteq L_m(A) \cap L_m(G)$.

It remains to show that $L(A) \cap L(G) \subseteq \overline{K}$ and $L_m(A) \cap L_m(G) \subseteq K$.

- *Proof of* $L(A) \cap L(G) \subseteq \overline{K}$.
  We show the inclusion $L(A) \cap L(G) \subseteq \overline{K}$ by induction on the length of strings.
  - *Base:* It is obvious that $\varepsilon \in (L(A) \cap L(G)) \cap \overline{K}$.
  - *Inductive Hypothesis:* Assume that $(\forall s \in \Sigma^*), |s| = n$ for some $n \geq 0$, $s \in L(A) \cap L(G) \Rightarrow s \in \overline{K}$. Now, we must show that $(\forall \sigma \in \Sigma)$ and $(\forall s \in \Sigma^*)$, $|s| = n$, $s\sigma \in L(A) \cap L(G) \Rightarrow s\sigma \in \overline{K}$. We proceed as follows:
    * Let $t = P(s)$. By Lemma 1, since $s\sigma \in L(A)$, there exists $s' \in \overline{K}$ such that $s'\sigma \in \overline{K}$ and $P(s') = t$.
    * Since K is Σ_{rel}-relevant w.r.t G, by Definition 3, the conditions $P(s) = P(s'), s'\sigma \in \overline{K}, s \in \overline{K}$ and $s\sigma \in L(G)$ together imply that $s\sigma \in \overline{K}$, validating the inductive hypothesis.

  Thus $L(A) \cap L(G) \subseteq \overline{K}$ and therefore $L(A) \cap L(G) = \overline{K}$.
- *Proof of* $L_m(A) \cap L_m(G) \subseteq K$.
  Assume that $s \in L_m(A) \cap L_m(G)$. We need to

show that $s \in K$, as follows.

- Since $L_m(A) \cap L_m(G) \subseteq L(A) \cap L_m(G)$, $s \in L(A) \cap L(G) = \overline{K}$ or $s \in \overline{K} \cap L_m(G)$.
- Let $t = P(s)$. By Lemma 1, since $s \in L_m(A)$, there exists $s' \in K$ such that $P(s') = t$.
- Since K is Σ_{rel}-relevant w.r.t G, by Definition 3, the conditions $P(s) = P(s'), s' \in K$ and $s \in \overline{K} \cap L_m(G)$ together imply that $s \in K$.

Thus $L_m(A) \cap L_m(G) \subseteq K$ and therefore $L_m(A) \cap L_m(G) = K$.

(Only If:) Let $A = (X, E, \xi, x_0, X_m)$ be a specification automaton of Definition 1 for G that is Σ_{irr}-transparent. It follows that $E = \Sigma$, and modeling K on G, $L(A) \cap L(G) = \overline{K}$ and $L_m(A) \cap L_m(G) = K$. We must then show that K is Σ_{rel}-relevant w.r.t G by establishing the two conditions of Definition 3.

Let $s, s' \in \Sigma^*$ such that $P(s) = P(s')$.

1) *Proof of Condition 1*: Let σ be an event in Σ such that $s\sigma \in \overline{K}$, $s'\sigma \in L(G)$ and $s' \in \overline{K}$. We then need to show that $s'\sigma \in \overline{K}$, as follows:
   - Because $s, s' \in \overline{K} \subseteq L(A)$ and $P(s) = P(s')$ and A is Σ_{irr}-transparent, A will be in the same state x after the execution of s and s', i.e., $\xi(s, x_0) = \xi(s', x_0) = x$.
   - Since $s\sigma \in \overline{K} \subseteq L(A)$, $\xi(\sigma, x)!$. Therefore $s'\sigma \in L(A)$. Thus, $s'\sigma \in L(A) \cap L(G) = \overline{K}$. Hence Condition 1 of Definition 3.
2) *Proof of Condition 2*: Assume that $s \in K$ and $s' \in \overline{K} \cap L_m(G)$. We then need to show that $s' \in K$, as follows:
   - Similar to the proof of Condition 1 above, because $s, s' \in \overline{K} \subseteq L(A)$ and $P(s) = P(s')$ and A is Σ_{irr}-transparent, A will be in the same state x after the execution of s and s', i.e., $\xi(s, x_0) = \xi(s', x_0) = x$.
   - Furthermore, since $s \in K \subseteq L_m(A)$, $x \in X_m$. Thus, $s' \in L_m(A)$. Therefore, $s' \in L_m(A) \cap L_m(G) = K$. Hence Condition 2 of Definition 3.

Thus by Definition 3, K is Σ_{rel}-relevant w.r.t G. ■

Corollary 1: Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, an automaton H representing a language $K \subseteq L_m(G)$, and an event subset $\Sigma_{irr} \subseteq \Sigma$. If K is $(\Sigma - \Sigma_{irr})$-relevant w.r.t G, then $A = Trans(H, \Sigma_{irr})$ is a specification automaton for G that is Σ_{irr}-transparent and that models K on G, i.e., $L_m(A) \cap L_m(G) = K$.

Proof: Immediate from the proof of the *If* statement in Theorem 1. ■

V. Procedures and Solution Algorithm

Theorem 1 has established an important connection between the concepts of a transparent specification automaton and a relevant specification language. From this theorem, it is clear that a maximally transparent specification automaton A modeling a specification language K for G can be synthesized from a relevant event subset Σ_{rel} for K of minimal cardinality (among all the event subsets that are relevant for K w.r.t G). A procedure to compute a minimal relevant event subset for K is, therefore, essential for computing a solution for Problem 1. Such a procedure is presented in this section (Section V-B). The procedure utilizes another procedure (Section V-A) to check for language relevance. In Section V-C, a provably correct solution algorithm for the main problem (Problem 1) is then presented.

A. Verification of Language Relevance

We first present a procedure to verify whether a language $K \subseteq L_m(G)$ is Σ_{rel}-relevant w.r.t a given DES G. Let A be a trim automaton that represents K, i.e., $L_m(A) = K$. Procedure $CheckRelevance$ returns $True$ if $L_m(A)$ is Σ_{rel}-relevant w.r.t G and $False$, otherwise.

Intuitively, $CheckRelevance$ builds automaton $RelTest(A, G)$ to track pairs of strings s and s' in $L(A)$, with $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s')$, and to determine the state of G reached after the execution of s'. Therefore, each state of $RelTest(A, G)$ is represented by a triple $(x, x', q) \in (X \times X \times Q)$, where $x, x' \in X$ are the states reached in A from x_0 after the execution of s and s', and $q \in Q$ is the state reached in G from q_0 after the execution of s'. Moreover, automaton $RelTest(A, G)$ also includes a special state called $dump \notin X \times X \times Q$ and a special event called $\gamma \notin \Sigma$ that are used to capture all violations of Σ_{rel}-relevance.

At any state (x, x', q) of $RelTest(A, G)$, if the occurrence of an event $\sigma \in \Sigma$ creates a violation of Condition 1 of Σ_{rel}-relevance (Definition 3), then a σ-transition from (x, x', q) to $dump$ is added to $RelTest(A, G)$. Furthermore, if the reach of a state (x, x', q) of $RelTest(A, G)$ creates a violation of

Condition 2 of Σ_{rel}-relevance, then a γ-transition from (x, x', q) to $dump$ is added to $RelTest(A, G)$. It is clear from the pseudo-code and the foregoing discussion that $CheckRelevance$ has polynomial time complexity of $O(|X|^2|Q|)$ where $|X|$ and $|Q|$ are the state size of A and G, respectively.

Procedure $CheckRelevance(A, G, \Sigma_{rel})$

```
Input: DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, specification
       automaton $A = (X, \Sigma, \xi, x_0, X_m)$ with
       $L_m(A) \subseteq L_m(G)$ and an event subset $\Sigma_{rel} \subseteq \Sigma$;
Output: $True$, if $L_m(A)$ is $\Sigma_{rel}$-relevant w.r.t $G$;
        $False$, otherwise;
begin
    Let $\Sigma_{irr} = \Sigma - \Sigma_{rel}$ and $\gamma$ be an event not in $\Sigma$;
    Step 1: Construct automaton $RelTest(A, G) = Ac((X \times X \times Q) \cup \{dump\}, \Sigma \cup \{\gamma\}, f, (x_0, x_0, q_0), \{dump\})$
    from $A$ and $G$ with the transition function $f$ defined as follows:
    $(\forall (x, x', q) \in X \times X \times Q)$:
    1) $(\forall \sigma \in \Sigma_{rel})$
        • $f(\sigma, (x, x', q)) = (\xi(\sigma, x), \xi(\sigma, x'), \delta(\sigma, q))$
          if $\xi(\sigma, x)!$, $\xi(\sigma, x')!$ and $\delta(\sigma, q)!$; and
        • $f(\sigma, (x, x', q)) = dump$
          if $\xi(\sigma, x)!$, $\neg\xi(\sigma, x')!$ and $\delta(\sigma, q)!$
    2) $(\forall \sigma \in \Sigma_{irr})$
        • $f(\sigma, (x, x', q)) = (\xi(\sigma, x), x', q)$
          if $\xi(\sigma, x)!$, $\neg\xi(\sigma, x')!$ and $\neg\delta(\sigma, q)!$; and
        • $f(\sigma, (x, x', q)) = (x, \xi(\sigma, x'), \delta(\sigma, q))$
          if $\neg\xi(\sigma, x)!$, $\xi(\sigma, x')!$ and $\delta(\sigma, q)!$; and
        • $f(\sigma, (x, x', q)) = dump$
          if $\xi(\sigma, x)!$, $\neg\xi(\sigma, x')!$ and $\delta(\sigma, q)!$
    3) $f(\gamma, (x, x', q)) = dump$ if $x \in X_m$, $x' \notin X_m$ and $q \in Q_m$.
    Step 2: Determine whether $L_m(A)$ is $\Sigma_{rel}$-relevant w.r.t $G$:
        • If $L_m(RelTest(A, G)) \neq \emptyset$, i.e., $dump$
          is encountered during the construction of
          $RelTest(A, G)$ in Step 1, return $False$;
        • Otherwise, return $True$;
```

Theorem 2: Given DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, specification automaton $A = (X, \Sigma, \xi, x_0, X_m)$ with $L_m(A) \subseteq L_m(G)$ and an event subset $\Sigma_{rel} \subseteq \Sigma$. Then $L_m(A)$ is Σ_{rel}-relevant w.r.t G if and only if $CheckRelevance(A, G, \Sigma_{rel}) = True$.

Proof: Let $RelTest(A, G)$ be the automaton constructed in Step 1 of $CheckRelevance$. We will prove this theorem by showing that $L_m(A)$ is Σ_{rel}-relevant w.r.t G if and only if $L_m(RelTest(A, G)) = \emptyset$.

To begin with, by construction of $RelTest(A, G)$ in Step 1 of $CheckRelevance$, it can be seen that a state triple $(x, x', q) \in X \times X \times Q$ is reachable in automaton $RelTest(A, G)$ if and only if there exists a pair of strings $(s, s') \in \overline{K} \times \overline{K}$ such that:

1) $\xi(s, x_0) = x$, $\xi(s', x_0) = x'$ and $\delta(s', q_0) = q$.
2) $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s')$.

(If:) Assume that $L_m(RelTest(A, G)) \neq \emptyset$, i.e., state $dump$ is reached by a σ-transition ($\sigma \in \Sigma$) or a γ-transition from some reachable state $(x, x', q) \in X \times X \times Q$ of $RelTest(A, G)$, we show that $L_m(A)$ is not Σ_{rel}-relevant w.r.t G, as follows.

Let $s, s' \in \overline{K}$ be any two strings with (1) $\xi(s, x_0) = x$, $\xi(s', x_0) = x'$ and $\delta(s', q_0) = q$; and (2) $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s')$. By construction of $RelTest(A, G)$, we have:

- If $dump$ is reached from (x, x', q) by a σ-transition for some $\sigma \in \Sigma$, then $s\sigma \in \overline{K}$, $s'\sigma \in L(G)$ and $s'\sigma \notin \overline{K}$, i.e., Condition 1 of Definition 3 is violated.
- If $dump$ is reached from (x, x', q) by a γ-transition, then $s \in K$, $s' \in \overline{K} \cap L_m(G)$ and $s' \notin K$, i.e., Condition 2 of Definition 3 is violated.

Thus, in either case, $L_m(A)$ is not Σ_{rel}-relevant w.r.t G.

(Only if:) Conversely, if $L_m(A)$ is not Σ_{rel}-relevant w.r.t G, there exists two strings $s, s' \in \overline{K}$ with $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s')$ such that:

1) $(\exists \sigma \in \Sigma)$ $s\sigma \in \overline{K}$, $s'\sigma \in L(G)$ and $s'\sigma \notin \overline{K}$; or
2) $s \in K$, $s' \in \overline{K} \cap L_m(G)$ and $s' \notin K$.

Let $x = \xi(s, x_0)$, $x' = \xi(s', x_0)$ and $q = \delta(s', q_0)$. Then $(x, x', q) \in X \times X \times Q$ is a reachable state of $RelTest(A, G)$.

By the construction of $RelTest(A, G)$ in Step 1 of $CheckRelevance$, if $(\exists \sigma \in \Sigma)$ $s\sigma \in \overline{K}$, $s'\sigma \in L(G)$ and $s'\sigma \notin \overline{K}$, state $dump$ will be reached from (x, x', q) via a σ-transition.

On the other hand, if $s \in K$, $s' \in \overline{K} \cap L_m(G)$ and $s' \notin K$, state $dump$ will be reached from (x, x', q) via a γ-transition.

Thus, in either case, state $dump$ is reachable in automaton $RelTest(A, G)$. Therefore $L_m(RelTest(A, G)) \neq \emptyset$.

■

Remark 2: Note that $Trans$ and $CheckRelevance$ are algorithmically similar to the respective procedures for computing a partially observable supervisor and checking language observability [18]. This similarity is not unexpected due to Remark 1.

B. Minimal Cardinality of Relevant Event Set

Lemma 2: Given DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, a language $K \subseteq L_m(G)$ and an event subset $\Sigma_{rel} \subseteq \Sigma$. If K is not Σ_{rel}-relevant w.r.t G then $(\forall \Sigma'_{rel} \subseteq \Sigma_{rel})$ K is not Σ'_{rel}-relevant w.r.t G.

Proof: By contradiction, assume that there exists some event subset $\Sigma'_{rel} \subseteq \Sigma_{rel}$ such that K is Σ'_{rel}-relevant w.r.t G. Then, since $(\forall s, s' \in \Sigma^*)$ $P_{\Sigma,\Sigma_{rel}}(s) = P_{\Sigma,\Sigma_{rel}}(s') \Rightarrow P_{\Sigma,\Sigma'_{rel}}(s) = P_{\Sigma,\Sigma'_{rel}}(s')$, it is easy to see that K is also Σ_{rel}-relevant w.r.t G, violating the assumption that K is not Σ_{rel}-relevant w.r.t G. ■

A relevant event set of minimal cardinality would result in making as many irrelevant events transparent as possible. Following our previous arguments, such an automaton should be the most preferable for better understandability among all available automata representing the same specification for a given DES.

Given DES G and a specification automaton A, a procedure called $MinRelevantSet$ is developed to compute a minimal (cardinality) event subset $\Sigma_{rel,min} \subseteq \Sigma$ such that $L_m(A)$ is $\Sigma_{rel,min}$-relevant w.r.t G. In essence, the procedure considers all subsets of Σ and selects from them a minimal subset $\Sigma_{rel,min}$ for which the relevance of $L_m(A)$ w.r.t G holds.

Procedure $MinRelevantSet$ uses a variable called $\Sigma_{rel,min}$ to store the minimal cardinality subset that has been found so far. It also uses a variable called $IrrelevantSets$ to store all the event subsets that are not qualified as relevant event subsets (for $L_m(A)$ w.r.t G).

Initially, $\Sigma_{rel,min} = \Sigma$ and $IrrelevantSets = \emptyset$. $MinRelevantSet$ starts by setting an index variable n to $|\Sigma| - 1$ and generating all subsets of Σ that have the cardinality of n. It then performs relevance checks for these generated event subsets by calling $CheckRelevance$.

Because of Lemma 2, upon determining that $L_m(A)$ is not Σ'-relevant, $MinRelevantSet$ adds all subsets of Σ' to the set of irrelevant event subsets $IrrelevantSets$, to avoid checking these subsets in future steps.

After checking all event subsets of cardinality n, the procedure decreases n by 1 and continues to generate and check all event subsets of cardinality $n-1$, to search for relevant event subsets of smaller cardinality and update $\Sigma_{rel,min}$ accordingly. During its search process, Procedure $MinRelevantSet$ maintains a variable called $count$ to count the number of event subsets of cardinality n that are not qualified as relevant event subsets. Whenever $count = \binom{|\Sigma|}{n}$, i.e., all the subsets of cardinality n or smaller are not qualified as relevant event subsets, or n reaches 0, the procedure stops and returns the relevant event subset of minimal cardinality that has been found.

Procedure $MinRelevantSet(G, A)$

```
Input: DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ and specification
       automaton $A = (X, \Sigma, \xi, x_0, X_m)$ with
       $L_m(A) \subseteq L_m(G)$;
Output: A minimal cardinality event subset
        $\Sigma_{rel,min} \subseteq \Sigma$ such that $L_m(A)$ is
        $\Sigma_{rel,min}$-relevant w.r.t $G$;
begin
    $IrrelevantSets \leftarrow \emptyset$;
    $\Sigma_{rel,min} \leftarrow \Sigma$;
    $n \leftarrow |\Sigma| - 1$; $OK = True$;
    while $OK = True$ and $n \geq 0$ do
        $count \leftarrow 0$;
        foreach $\Sigma' \subseteq \Sigma$ and $|\Sigma'| = n$ do
            if $\Sigma' \in IrrelevantSets$ then
                $count \leftarrow count + 1$;
            else if $CheckRelevance(G, A, \Sigma') = False$
            then
                $count \leftarrow count + 1$;
                Add $\Sigma'$ and all of its subsets to
                $IrrelevantSets$;
            else
                $\Sigma_{rel,min} \leftarrow \Sigma'$;
        if $count = \binom{|\Sigma|}{n}$ then $OK = false$;
        $n \leftarrow n - 1$;
    Return $\Sigma_{rel,min}$;
```

In the worst case, $MinRelevantSet$ has to examine all the (strict) subsets of Σ for language relevance, and as a result, it has to call $CheckRelevance$ $2^{|\Sigma|} - 1$ times. Therefore, $MinRelevantSet$ has exponential time complexity of $O(2^{|\Sigma|}|X|^2|Q|)$, where $|X|$ and $|Q|$ are the state size of A and G, respectively. To speed up $MinRelevantSet$, a pruning technique based on Lemma 2 can be used. Specifically, after discovering that $L_m(A)$ is not relevant w.r.t $\Sigma' \subseteq \Sigma$, $MinRelevantSet$ can store all the subsets of Σ' into a data structure, and avoid checking for language relevance w.r.t these subsets in future steps.

When computational time is expensive, however, an algorithm of polynomial time complexity is of practical interest. To avoid searching all the subsets of Σ, and hence reduce the computational time, such an algorithm may compute and return a relevant event set with reasonably small (but not necessary minimal) cardinality for $L_m(A)$ w.r.t G. However, the development of such an algorithm is beyond the scope of this paper.

C. Solution Algorithm

In what follows, Algorithm 1 is proposed for computing a specification automaton as a solution for Problem 1. The algorithm has two main steps: (1) it computes a maximal set of irrelevant events using $MinRelevantSet$; and (2) it uses the computed event set to synthesize the solution specification automaton using $Trans$.

Let $|Y|$ and $|Q|$ be the state size of H and G, respectively. Since Algorithm 1 is built on the foundation of the two procedures $MinRelevantSet$ and $Trans$, it has time complexity of $O((2^{|\Sigma|}-1)|Y|^2|Q|+2^{|Y|})$, which is the “summation” of their time complexities.

In practice, the computational complexity of Algorithm 1 might need to be reduced to deal with large systems. In doing so, the complexities of the individual procedures $MinRelevantSet$ and $Trans$ would need to be mitigated. An approach to reduce the computational complexity of $MinRelevantSet$ has been discussed in Section V-B. In what follows, we discuss how the computational complexity of Procedure $Trans$ can be reduced. In Step 2 of Algorithm 1, $Trans$ is invoked to compute specification automaton $A = Trans(H, \Sigma_{irr,max})$. As pointed out in Section IV, the exponential complexity of $Trans$ is due to the projection of automaton H onto event subset $\Sigma_{rel,min} = \Sigma - \Sigma_{irr,max}$. This exponential complexity can be avoided if H has some special structure w.r.t $\Sigma_{rel,min}$. For instance, if the natural projection $P_{\Sigma,\Sigma_{rel,min}}$ is an observer of $L_m(H)$ [19], i.e., $(\forall t \in P_{\Sigma,\Sigma_{rel,min}}(L_m(H)))(\forall s \in L(H))$ $[P_{\Sigma,\Sigma_{rel,min}}(s)$ is a prefix of $t] \Rightarrow (\exists u \in \Sigma^*)[su \in L_m(H)$ and $P_{\Sigma,\Sigma_{rel,min}}(su) = t]$, then the projected image of H onto $\Sigma_{rel,min}$ can be computed in polynomial time [19]; and it would follow that $Trans$ has polynomial time complexity. Thus, to reduce the computational complexity of $Trans$, event subset $\Sigma_{rel,min}$ could be enlarged, if necessary, to satisfy the observer condition [19]. By Lemma 2, this set enlargement does not violate the relevance property of $L_m(H)$ w.r.t G. However, one should note that enlarging $\Sigma_{rel,min}$ this way means that the maximal cardinality of irrelevant event set $\Sigma_{irr,max} = \Sigma - \Sigma_{rel,min}$ is no longer guaranteed, and for this reason, the output automaton A may not be maximally transparent.

Algorithm 1: Maximally transparent specification automaton synthesis

```
Input: DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ and an automaton $H$ with
    $L_m(H) = K \subseteq L_m(G)$;
Output: Specification automaton $A$ that models $K$ on $G$ and
    has a maximal cardinality set of irrelevant events;
begin
    Step 1: Compute a maximal cardinality set of irrelevant
    events:
        • Step 1.a: $\Sigma_{rel,min} = MinRelevantSet(G, H)$;
        • Step 1.b: $\Sigma_{irr,max} = \Sigma - \Sigma_{rel,min}$;
    Step 2: Compute a $\Sigma_{irr,max}$-transparent automaton $A$
    that models $K$ on $G$: $A = Trans(H, \Sigma_{irr,max})$;
    Return $A$;
```

Theorem 3: With $L_m(H) = K$, Algorithm 1 returns a solution automaton for the transparency maximization Problem 1.

Proof: Let $\Sigma_{rel,min}$ and $\Sigma_{irr,max} = \Sigma - \Sigma_{rel,min}$ be the event subsets generated in Steps 1.a and 1.b of Algorithm 1, and $A = Trans(H, \Sigma_{irr,max})$ be the automaton generated in Step 2 of Algorithm 1. It is clear that K is $\Sigma_{rel,min}$-relevant w.r.t G. Therefore, according to Corollary 1, A is a specification automaton that is $\Sigma_{irr,max}$-transparent and that models K on G.

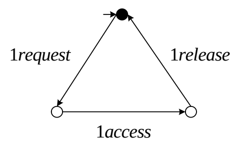
Also, since $\Sigma_{rel,min}$ is a minimal relevant event set for K w.r.t G, there is no specification automaton that models K on G and has a set of irrelevant events with a greater cardinality than $|\Sigma_{irr,max}|$: if there is such a Σ'-transparent specification automaton with $|\Sigma'| > |\Sigma_{irr,max}|$, then $|\Sigma - \Sigma'| < |\Sigma_{rel,min}|$, and according to Theorem 1, K is $(\Sigma - \Sigma')$-relevant w.r.t G, contradicting the fact that $\Sigma_{rel,min}$ is a relevant event set with minimal cardinality for K w.r.t G.

Thus, Algorithm 1 generates specification automaton A that models K on G and has a maximal set of irrelevant events, i.e., Algorithm 1 synthesizes a solution automaton for Problem 1.

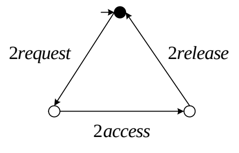
■

VI. Illustrative Examples

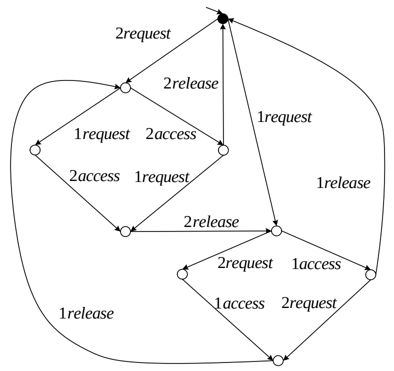
We now present two examples to illustrate the concept of a maximally transparent specification automaton. In the illustration, every automaton is shown as a directed graph with the initial state represented by a node with an entering arrow, and every marker state represented by a darkened node.



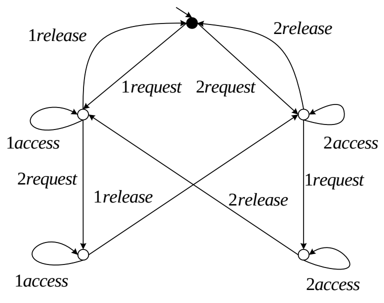
(a) $USER_1$



(b) $USER_2$



(c) FCFS specification automaton H



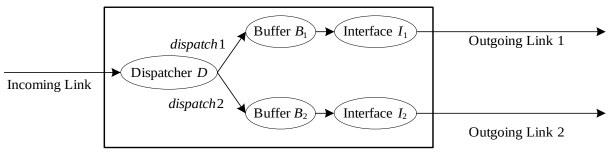
(d) Maximally transparent FCFS specification

Fig. 1. Illustrative example 1: Resource allocation system

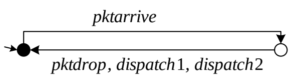
Example 1 (Resource allocation): The first example is a first come, first served (FCFS) control requirement for a resource allocation system. The example system model, denoted by G, is a synchronous product [18] of two users, $USER_1$ [Fig. 1(a)] and $USER_2$ [Fig. 1(b)]. The automaton $USER_i$, $i \in \{1,2\}$, is modeled to request, access and release a resource; and the system G models their asynchronous operations to share the single resource.

A specification automaton H of the FCFS requirement for G is shown in Fig. 1(c). It is due to some specification automaton P prescribed by a system designer such that $L_m(P) \cap L_m(G) = L_m(H)$. Applying Algorithm 1 to H, we obtain a maximally transparent specification automaton as shown in Fig. 1(d), with $\Sigma_{irr,max} = \{1access, 2access\}$.

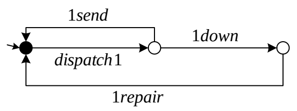
Observe that the events in $\Sigma_{irr,max}$ appear only in self-loops. Importantly, for $i, j \in \{1,2\}$, that i$release$ precedes j$release$ whenever i$request$ precedes j$request$ - the essence of FCFS for G - is quite clearly highlighted in the resulting automaton of Fig. 1(d), but may not be as obvious in Fig. 1(c) or some other specification automaton P prescribed by the system designer.



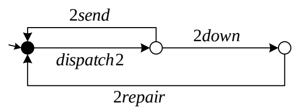
(a) System layout



(b) Dispatcher D



(c) Interface I_1



(d) Interface I_2

Fig. 2. Illustrative example 2: Network switching system

Example 2 (Computer network switch): The second example is a computer network switching system. In this simplified system, a network switch is modeled to have one incoming link and two outgoing links [Fig. 2(a)]. The example system model, denoted by S, is a synchronous product of a dispatcher D [Fig. 2(b)] and two network interfaces I_1 [Fig 2(c)] and I_2 [Fig. 2(d)]. The system works as follows. When a data packet arrives at the incoming link, the dispatcher either drops the packet or deposits it into one of the two buffers B_1 and B_2 for the respective outgoing interfaces. When a packet is deposited into a buffer, the corresponding network interface can send the packet to the outgoing link connected to it unless it breaks down, in which case it would need to be repaired. Upon repairing and resuming operation from a breakdown, a network interface's buffer becomes empty, meaning that every packet deposited into the buffer before the interface broke down is lost.

Consider a network switch (NS) control requirement, informally stated as follows: The buffers must never overflow or underflow, and if both the interfaces break down, interface I_1 must be

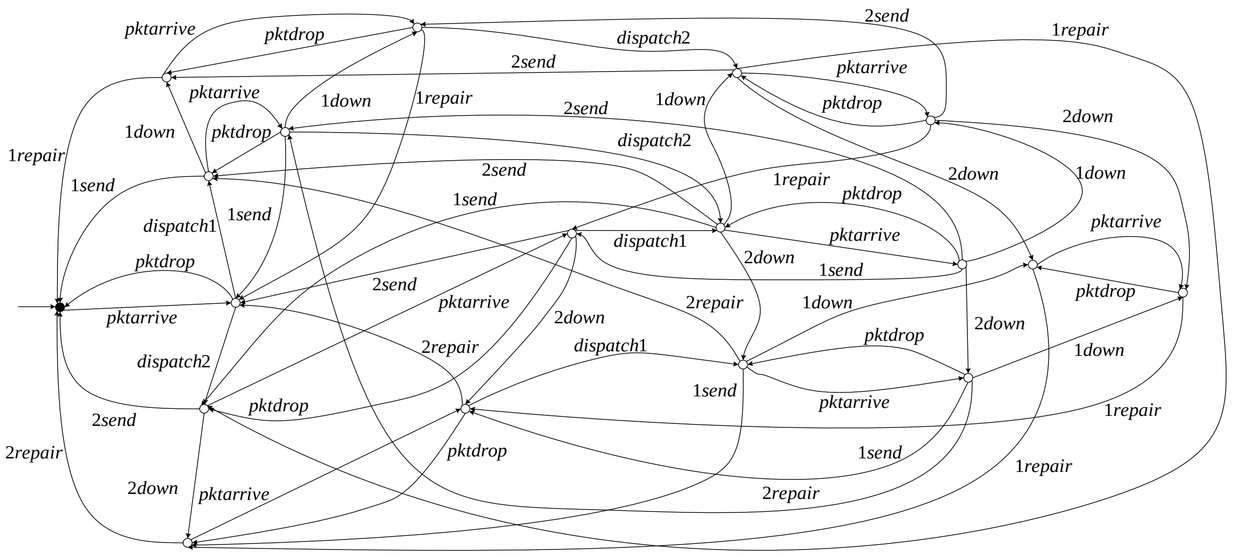


Fig. 3. NS specification automaton H

repaired first. Assume that B_1 and B_2 are one-slot buffers. Then more formally, for $i \in \{1, 2\}$, the NS specification requires $dispatch$i to be executed first, and i$send$ (or i$repair$) and $dispatch$i to be executed alternately thereafter; and if $1down$ occurs, then $2repair$ cannot be executed until $1repair$ is. This requirement can be modeled by a specification automaton H as shown in Fig. 3. This automaton is prescribed by carefully considering every possible event string generated by the synchronous product of the three automata in Figs. 2(b)-2(d) that form the system model S, and excluding every of those that violate the control requirement. Nevertheless, when presented with this specification automaton, it is hard for a designer to comprehend the specification and readily ascertain if the automaton models the intended requirement.

Applying Algorithm 1, we obtain a maximally transparent specification automaton as shown in Fig. 4, with $\Sigma_{irr,max} = \{pktarrive, pktdrop, 2down\}$. Clearly, the essence of the NS specification for the system S is expressed more evidently in Fig. 4 than in Fig. 3, demonstrating the utility of our algorithm.

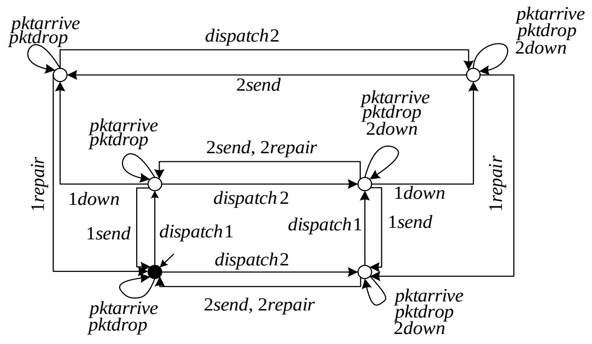


Fig. 4. Maximally transparent NS specification

VII. CONCLUSION

We have motivated and developed the notion of transparency of automata as specifications for DES. We have formalized the transparency maximization problem and developed an algorithm to construct a maximally transparent specification automaton. Two illustrative examples show that such a specification automaton can improve designer comprehensibility by better highlighting the essential precedence order of only those relevant events that collectively constitute the essence of the specification.

To harness the specifiability and readability of temporal logic in an automata-based DES framework, previous work [6] has proposed an algorithm that translates a (finitary) temporal logic specification to a specification automaton H generating a sublanguage of $L_m(G)$ for a given DES G. An interesting avenue for future research is to translate such a temporal logic specification directly to a maximally transparent specification automaton A for which $L_m(H) = L_m(A) \cap L_m(G)$, without explicitly constructing H. Together, this could promote a more effective specification-synthesis paradigm, where the natural language readability of a temporal logic specification and the transparency of the translated automaton A could render higher confidence that the specification automaton - a mandatory input for control synthesis of DES using automata-based tools - does indeed capture the intended requirement.

To deal with large control systems, it is also worthwhile to develop complexity reduction strategies for Algorithm 1 that leverage on recent advancement in finite automata.

Finally, beyond the DES domain, our research may be applicable in other domains where automata are used for modeling and specification.

REFERENCES

[1] M. T. Pham, A. Dhananjayan, and K. T. Seow, "On the transparency of automata as discrete-event control specifications," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010, pp. 1474–1479.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, January 1987.

[3] E. Roszkowska, "Supervisory control for deadlock avoidance in compound processes," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 34, no. 1, pp. 52–64, 2004.

[4] J. Huang and R. Kumar, "An optimal directed control framework for discrete event systems," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 37, no. 5, pp. 780 – 791, 2007.

[5] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA : Addison-Wesley, 1979.

[6] K. T. Seow, "Integrating temporal logic as a state-based specification language for discrete-event control design in finite automata," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 451–464, July 2007.

[7] S. L. Ricker, N. Sarkar, and K. Rudie, "A discrete event systems approach to modeling dextrous manipulation," *Robotica*, vol. 14, no. 5, pp. 515–525, 1996.

[8] S. C. Lauzon, A. K. L. Ma, J. K. Mills, and B. Benhabib, "Application of discrete event system theory to flexible manufacturing," *IEEE Control Systems Magazine*, vol. 16, no. 1, pp. 41–48, February 1996.

[9] R. G. Qiu and S. B. Joshi, "A structured adaptive supervisory control methodology for modeling the control of a discrete event manufacturing system," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 29, no. 6, pp. 573 – 586, 1999.

[10] S. Wang, S. F. Chew, and M. A. Lawley, "Using shared-resource capacity for robust control of failure-prone manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 38, no. 3, pp. 605 – 627, 2008.

[11] K. T. Seow and M. Pasquier, "Supervising passenger land-transport systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 3, pp. 165–176, September 2004.

[12] W. M. Wonham, "Supervisory control theory: Models and method," in *Proceedings of the 24th International Conference on Application Theory of Petri Nets*, Eindhoven, The Netherlands, June 2003, pp. 1–14.

[13] K. T. Seow and R. Devanathan, "A temporal logic approach to discrete event control for the safety canonical class," *Systems and Control Letters*, vol. 28, no. 4, pp. 205–217, 1996.

[14] S. Miremadi, K. Akesson, and B. Lennartson, "Extraction and representation of a supervisor using guards in extended finite automata," in *Proceedings of the 9th International Workshop on Discrete Event Systems*, Gothenburg, Sweden, May 2008, pp. 193–199.

[15] R. Su and W. M. Wonham, "Supervisor reduction for discrete-event systems," *Discrete Event Dynamic Systems : Theory and Applications*, vol. 14, no. 1, pp. 31–53, 2004.

[16] S.-J. Whittaker and K. Rudie, "Lose fat, not muscle: An examination of supervisor reduction in discrete-event systems," *Discrete Event Dynamic Systems*, vol. 18, no. 3, pp. 285–321, 2008.

[17] F. Lin and W. M. Wonham, "On observability of discrete event systems," *Information Sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[18] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2008.

[19] L. Feng and W. M. Wonham, "Supervisory control architecture for discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1449–1461, July 2008.