doi:10.1093/comjnl/bxh000

Testing as Collecting of Evidence: An Integrated Approach to Test Generation for Finite State Machines

Monika Kapus-Kolar

Jožef Stefan Institute, Ljubljana, Slovenia
Email: monika.kapus-kolar@ijs.si

A new method for generating tests for deterministic implementations of deterministic finite state machines is proposed. The method is generic, taking as a parameter a wide class of testing strategies and able to support both transition-oriented and fault-model-driven testing. For a given specification machine, it builds a graph encoding the given strategy and then generates a test by solving on the graph a generalization of the Rural Postman Problem. Only the first phase is domain-specific and therefore precisely described in the paper. The second phase is considered a responsibility of graph specialists, though we provide guidelines for solving the most commonly encountered special case. The strategy encoding produced in the first phase is such that the second phase automatically handles, in an integrated manner, the following optimization concerns: absolute avoidance of forbidden transitions, maximum avoidance of transitions which are for some reason considered undesirable, subtest choice, subtest ordering and subtest connection, possibly with overlapping. The method accepts multi-criteria transition cost functions. If both its phases are executed precisely, it generates a test optimal with respect to the adopted strategy, and a strategy for generating absolutely optimal tests is also given. For the cases where the complexity of the first phase or of the resulting graph is problematic, guidelines for systematically conducting the phase in an approximate way are provided.

Received 16 May 2006; revised 4 October 2006; accepted 14 November 2006

The paper discusses generation of tests for systems which can be abstractly described as finite state machines. From the *practitioners' perspective*, it is a paper taking some of the best-known test generation methods and polishing them to perfection, so that they start *generating tests which are optimal*, in the sense that they satisfy the adopted testing strategy as much as possible and at a minimum cost. However, the paper should be understood primarily as a contribution towards a *broader perspective on automated test generation* and as an initiative for a *paradigm shift*.

The work has been motivated primarily by the observation that the test generation method of [1] pursues a very reasonable testing strategy, but fails to implement it precisely, consequently generating tests which are slightly longer than expected. The discrepancy between the method of [1] and the strategy which it presumably tries to implement is very tiny; so tiny that it cannot be perceived unless one reconsiders the test generation process down to its conceptual essence. Consequently, our development of an exact implementation of the strategy has led to development of a *new, generic test generation method based on a conceptual framework in which testing strategies and algorithms for generating tests reflecting the strategies are two clearly separated concepts*.

The new approach is very convenient, both for analyzing and comparing the existing test generation methods and for developing new ones. When employed in method analysis and comparison, it nicely supports *separate consideration of the qualitative and the quantitative aspects of the test generation process*. When employed in method synthesis, its advantage lies in the fact that *strategy specification can be done in a purely declarative manner, while strategy implementation can be an automated procedure accepting a range of different strategies*.

To convey the ideas, the paper describes not only improvements over [1] and some other well-known test

generation methods, but also the generic method itself, presenting the improved specific methods just as its example specializations.

1. INTRODUCTION

A *finite state machine* (FSM) is a machine with a root state and with no more than a finite number of other states, whose every transition is characterized by its initial state, the triggering input, the resulting output and the resulting state. In this paper, we consider the problem of generating tests for discriminating between an initially connected and deterministic specification FSM M and faulty implementation FSMs M' that are assumed to be defined on the same input and output alphabets as M, initially connected, deterministic and completely specified. For an M, let Φ_M denote the set of its expected faulty implementations. The probability of their occurrence is assumed to be unknown. An M' is considered faulty if there is a sequence of inputs (i.e., a test) to which both M and M' are able to react, but with different sequences of outputs. Here we assume that applying a test to an FSM means applying it to the root state.

We assume that it is acceptable to apply to an M' any test which would, if applied to M instead, trigger none of the transitions declared as forbidden. The usual reason for forbidding a transition is that it is actually non-existent, i.e., that it has been specified only because one wanted to make M completely specified. As it is desirable that testing leaves no evidence that it has been performed, we assume that any test, at least when applied to M itself, must terminate by returning the FSM to the root. As for the cost of executing a test on an M', we assume that it is the same as for its execution on M.

We propose a test construction method handling a wide range of optimization concerns in an *integrated* manner, thereby facilitating generation of optimal tests. The method is *generic*, accepting as a parameter a wide class of testing strategies, thereby facilitating systematic implementation of well-defined testing strategies. Another welcome feature of the method is *strict separation of domain-specific and general graph-theoretic concerns*, where we mainly focus on the question on how to reduce the test generation problem for a deterministic M (DTGP) to some generic graph-theoretic problem, i.e., to a problem under the responsibility of graph specialists. However, for the most commonly encountered graph-theoretic problem, we also suggest how it can be solved.

The method accepts *multi-criteria transition cost functions*, meaning that for every transition of a system, one is allowed to specify the cost associated with its execution from multiple different aspects, provided that one also specifies the relative importance of the aspects. For a transition, one could, for example, indicate its time consumption on the one hand and its energy consumption on the other, and then declare that for the sequence of transitions to be generated for the specified test purpose, the minimization of time consumption is more important than that of energy consumption, i.e., that the criterion of energy consumption is relevant only for choosing between sequences with the same time consumption.

If the method is executed precisely, it generates a test *optimal with respect to the adopted testing strategy*, i.e., a minimum-cost test among those satisfying the *acceptance criteria* of the strategy. A strategy for generating *absolutely optimal tests*, i.e., minimum-cost tests among those failing on every M' in Φ_M, is also given. For the cases where the complexity of the domain-specific phase of the method or of the resulting graph is problematic, guidelines for systematically conducting the phase in an approximate way are provided.

In our view, a *testing strategy* is a statement on what kind of an input/output (I/O) sequence $\bar{z}$ executable from the root state of M one should try to provoke on given implementations M', plus possibly some additional transition cost criteria. A $\bar{z}$ is a *candidate sequence*, i.e., satisfies the acceptance criteria of a specific strategy, if 1) it returns M to its root, 2) it provokes on M no forbidden transitions, and 3) according to the strategy, its observation on an M' provides sufficient evidence of the correctness of M'.

Among the (implicitly or explicitly) specified candidate sequences, there might be some which are not reliable witnesses of the correctness of M', for example, because the specifier of the strategy has lacked skills or time, or because she/he suspected that sequences checking M' more thoroughly would be too expensive to apply, and therefore deliberately made the third acceptance criterion less demanding. It is, for example, not unusual to apply a test for an individual transition without first checking that the machine is currently in the initial state of the transition [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. To compensate such carelessness, a specifier might want to introduce *additional cost criteria* working against transitions whose traversal is suspected to reduce the discriminating power of the test (cf. Section 5.6).

Our method is based on the observation that testing is by definition an activity pursuing the goal of *collecting evidence of correctness*. However, as testing strategies differ in the level and focus of scrutiny, the goal is too abstract to be operational. We observe that a typical strategy implicitly or explicitly refines the goal into one or more *alternative sets of test goals*, where it suffices to satisfy the goals in one of the sets. An individual *test goal* denotes collection of evidence that the tested M' possesses some specific properties.

As the FSM on which a test I/O sequence $\bar{z}$ is attempted is in the ideal case an exact copy of M, it is customary to interpret $\bar{z}$ as a *test tour* (TT) in M. A TT satisfies a test goal by including *a specific transition in a specific context*. For example, by executing a

transition τ from M in such a position in TT that the previously observed inputs and outputs verify the initial state of τ and the subsequently observed inputs and outputs verify its final state, one satisfies the goal of τ verification. There is often more than one way to satisfy a goal. For the just mentioned goal, any two I/O sequences respectively recognizing the two states would provide a satisfactory context for τ with respect to the goal.

A part which within a TT provides evidence that a specific goal is satisfied is a *subtest* implementing the goal. Our method handles the following *optimization concerns*: absolute avoidance of forbidden transitions, maximum avoidance of undesirable transitions, subtest choice, subtest ordering and subtest connection, possibly with overlapping.

To *formalize* a testing strategy, for each of its test goals one specifies the *predicate* that a TT must satisfy to satisfy the goal. These predicates, together with the auxiliary predicates in terms of which they are expressed, form the *predicate system* of the strategy. Predicate systems differ in how easily they can be manipulated during TT construction. Our method currently focuses on a specific class of predicate systems whose manipulation is particularly easy, but which are nevertheless sufficiently expressive for formalization of the usual kinds of testing strategies. We call such predicate systems *canonical*. By providing a *uniform encoding* for a large class of testing strategies, we establish a *unified conceptual framework* for the comparison of the existing strategies and for the definition of new ones.

It turns out that among the strategies for DTGP whose predicate system is canonical, there is also the one leading to an absolutely optimal TT. We show that it in a general case reduces a DTGP to an instance of a *non-standard generalization* GRPP *of the standard Rural Postman Problem* (RPP), and not to an RPP. Hence, if one follows the current practice of forcibly reducing every DTGP to the specialization RPP of GRPP, one typically makes some premature and non-optimal decisions and, consequently, produces a non-optimal TT. We also observe that many of the well-known methods for DTGP are actually approximate implementations of strategies pursuing many alternative goal sets, i.e., of strategies requiring that DTGP is reduced to G^{2}RPP, a *yet further non-standard generalization of RPP*.

The rest of the paper is organized as follows: Section 2 introduces the basic formal concepts and notations used in the paper, particularly those informally encountered above. In Section 3, we describe a very common and well investigated special case of DTGP which we call the *classical testing problem* (CTP), and use it to explain why we believe that time has come for a paradigm shift in the solving of DTGP.

In Section 4, we show how to reduce DTGP to G^{2}RPP, with GRPP denoting the special case with a single goal set. In Section 5, we discuss some applications of the reduction, among others, an exact implementation of the strategy of [1]. In Section 6, we reduce G^{2}RPP to a *non-standard generalization* G^{2}TSP *of the standard Generalized Travelling Salesman Problem* (generalized TSP, GTSP). We leave G^{2}TSP for further study, but point to some recent methods for GTSP, to which GRPP reduces.

As the proposed method encodes a given testing strategy for a given M by a graph whose size might be enormous, we in Section 7 suggest how to keep the size reasonable by including into the graph only selected parts of the available information on possibilities for test optimization. In Section 8, we make a final assessment of the contributions of the paper. In Section 9, we provide some suggestions for further work.

2. NOTATIONS AND DEFINITIONS

2.1. Sequences

A sequence (or a vector) $\bar{o}$ of some objects o_i is an $o_1, \ldots, o_k$. In an extreme case, its size k is 0. An empty sequence will be denoted by ε. Where we put an object o in a position where one by definition expects a sequence of objects of the class to which o belongs, it should be interpreted as a sequence with o the only component.

For a sequence $\bar{o}$, $cmps(\bar{o})$ denotes the set of its components, $cmp(\bar{o}, i)$ denotes its i-th component, $pfx(\bar{o}, i)$ its prefix of length i, and $sfx(\bar{o}, i)$ its suffix of length i. *Concatenation* $\bar{o}\cdot\bar{o}'$ of an $\bar{o} = o_1, \ldots, o_k$ and an $\bar{o}' = o'_1, \ldots, o'_{k'}$ is sequence $o_1, \ldots, o_k, o'_1, \ldots, o'_{k'}$.

2.2. Directed Graphs

A *directed graph (digraph)* G is defined by a tuple (V, E) in which V is a non-empty set of *vertices* and E is a set of directed *edges* between the vertices. If a vertex v_0 in V is designated as the root, G is *rooted* and more precisely defined by (V, v_0, E).

An edge e defined by a tuple $(v, v', l, \bar{c})$ goes from vertex v to vertex v'. l is a (possibly empty) sequence of some symbols and represents the *label* of e. $\bar{c}$ is the *cost vector* representing the cost of traversing e, with different components corresponding to *different cost criteria*, where the size of the vector is the same for every edge in G. We assume that every component of a cost vector is a non-negative real. A $\bar{c}$ of the expected size with all components 0 will be denoted by $\bar{0}$.

For *comparing cost vectors*, we define that $\bar{c} < \bar{c}'$ if there is a i such that for every $1 \leq j < i$, $cmp(\bar{c}, j) = cmp(\bar{c}', j)$, and $cmp(\bar{c}, i) < cmp(\bar{c}', i)$, i.e., the costs listed earlier are considered absolutely more important than those listed later.

A sequence $\bar{e} = e_1, \ldots, e_k$, $e_i = (v_i, v_{i+1}, l_i, \bar{c}_i)$ for $1 \leq i \leq k$, of consecutive edges of G is called a *walk* and has an *initial vertex* $init(\bar{e}) = v_1$, a *final vertex* $fin(\bar{e}) = v_{k+1}$, a *label* $lab(\bar{e}) = l_1 \cdot \ldots \cdot l_k$ and a *cost*

vector cost$(\bar{e}) = \bar{c}_1 + \ldots + \bar{c}_k$. For the extreme case of $\bar{e} = \varepsilon$, where $init(\bar{e})$ and $fin(\bar{e})$ cannot be assessed as above, we assume that the vertex $init(\bar{e}) = fin(\bar{e})$ is evident from the context. If $init(\bar{e}) = fin(\bar{e})$, $\bar{e}$ is *closed*. A closed walk is a *tour*. If $init(\bar{e}) = v_0$, $\bar{e}$ is *initially rooted*. If $fin(\bar{e}) = v_0$, $\bar{e}$ is *finally rooted*. A tour is *rooted* if it is initially rooted. If $\bar{e} = \varepsilon$ or if v_1 to v_k are distinct, $\bar{e}$ is a *path*. A closed path is a *circuit*. A circuit with $k = 1$ is a *loop*. Let $vrts(\bar{e})$ denote the set $\{v_1, \ldots, v_{k+1}\}$ if $\bar{e} \neq \varepsilon$, and $\{init(\bar{e})\}$ otherwise.

If there is a path from v_0 to every v in V, G is *initially connected*. If there is a path from every v in V to every v' in V, G is *strongly connected*.

A digraph (V', E') is a *subgraph* of a digraph (V, E) if $V' \subseteq V$ and $E' \subseteq E$.

2.3. Finite State Machines

An FSM M is defined by a tuple (S, s_0, X, Y, T) in which S is a finite set of *states*, s_0 is the *root state* in S, X is a finite *input alphabet*, Y is a finite *output alphabet* and T is a set of *transitions* $(s, s', x/y, \bar{c})$ with s and s' in S, x in X, y in Y and $\bar{c}$ a cost vector, so that M can also be understood as a rooted digraph (S, s_0, T).

If T comprises an $(s, \_, x/\_, \_)$ for every s in S and x in X, M is *completely specified*. If there exist such functions δ, λ and γ that every τ in T is an $(s, \delta(s,x), x/\lambda(s,x), \gamma(s,x))$, M is *deterministic*.

A sequence $\bar{\tau} = (s_1, s_{k+1}, \bar{z}, \bar{c}) = \tau_1, \ldots, \tau_k$, $\tau_i = (s_i, s_{i+1}, x_i/y_i, \bar{c}_i)$ for $1 \leq i \leq k$, of consecutive transitions of M is called a *transition sequence* and has an *initial state* $init(\bar{\tau}) = s_1$, a *final state* $fin(\bar{\tau}) = s_{k+1}$, a *label* $lab(\bar{\tau}) = \bar{z} = z_1, \ldots, z_k = x_1/y_1, \ldots, x_k/y_k$ with *input part* $in(\bar{\tau}) = in(\bar{z}) = x_1, \ldots, x_k$ and *output part* $out(\bar{\tau}) = out(\bar{z}) = y_1, \ldots, y_k$, and a *cost vector* $cost(\bar{\tau}) = \bar{c} = \bar{c}_1 + \ldots + \bar{c}_k$. For the extreme case of $\bar{\tau} = \varepsilon$, we assume that the state $init(\bar{\tau}) = fin(\bar{\tau})$ is evident from the context.

In an M, the *forward exclusion set* $FE(\bar{z})$ of an I/O sequence $\bar{z}$ is the set of all states s for which there is no $\bar{\tau}$ with $init(\bar{\tau}) = s$ and $lab(\bar{\tau}) = \bar{z}$. The *backward exclusion set* $BE(\bar{z})$ of a $\bar{z}$ is the set of all states s for which there is no $\bar{\tau}$ with $fin(\bar{\tau}) = s$ and $lab(\bar{\tau}) = \bar{z}$.

2.4. Tour Construction Problems on Digraphs

The directed *Travelling Salesman Problem* (TSP) is defined as follows [11]: Given a digraph $G = (V, E)$ with $E = \{(v, v', l_{v,v'}, \bar{c}_{v,v'}) | v, v' \in V, v \neq v'\}$, find in G a minimum-cost circuit $\bar{e}$ such that $vrts(\bar{e}) = V$.

The directed *Generalized TSP* (GTSP) is defined as follows [11]: Given G as for TSP, a non-empty set Γ of goals and a partition of V into non-empty, possibly intersecting subsets V_g for g in Γ, find in G a minimum-cost circuit $\bar{e}$ such that for every g in Γ, $vrts(\bar{e}) \cap V_g \neq \emptyset$. TSP corresponds to the special case with all V_g singleton.

Let the directed *Generalized GTSP* (G^2TSP) be defined as follow: Given G, Γ and V_g as for GTSP, and a partition Ξ of Γ into non-empty, possibly intersecting subsets, find in G a minimum-cost circuit $\bar{e}$ such that for some goal set ξ in Ξ, $vrts(\bar{e}) \cap V_g \neq \emptyset$ for every g in ξ. GTSP corresponds to the special case with Ξ singleton.

The directed *Rural Postman Problem* (RPP) is defined as follows [12]: Given a strongly connected digraph $G = (V, E)$ and an $\emptyset \subset E' \subseteq E$, find in G a minimum-cost tour $\bar{e}$ such that $E' \subseteq cmps(\bar{e})$.

Let the directed *Generalized RPP* (GRPP) be defined as follows: Given G as for RPP, a non-empty set Γ of goals and for every g in Γ an $\emptyset \subset E_g \subseteq E$, find in G a minimum-cost tour $\bar{e}$ such that for every g in Γ, $cmps(\bar{e}) \cap E_g \neq \emptyset$. RPP corresponds to the special case with all E_g singleton.

Let the directed *Generalized GRPP* (G^2RPP) be defined as follows: Given G, Γ and E_g as for GRPP, and a partition Ξ of Γ into non-empty, possibly intersecting subsets, find in G a minimum-cost tour $\bar{e}$ such that for some goal set ξ in Ξ, $cmps(\bar{e}) \cap E_g \neq \emptyset$ for every g in ξ. GRPP corresponds to the special case with Ξ singleton.

We adopt the following *interpretation*: The g in Γ are the *products* which one would potentially like to collect during the tour. The v in a V_g are the *warehouses* stocking the product g. The e in an E_g are the edges hosting a warehouse v_e (not represented in V) stocking g.

If for a G, one requires that the produced tour goes through its root v_0, so that it can be interpreted as starting in it, this must be specified as an additional goal g_0 in Γ. For GTSP or G^2TSP, one defines $V_{g_0} = \{v_0\}$. For GRPP or G^2RPP, one enhances G with a zero-cost dummy loop e_0 in v_0 and defines $E_{g_0} = \{e_0\}$. Besides, for G^2TSP or G^2RPP, g_0 must be included in every goal set ξ in Ξ.

2.5. Predicates on Rooted Walks of Digraphs

Solving a rooted tour construction problem for a strongly connected digraph $G = (V, v_0, E)$, one has to evaluate various *predicates on rooted tours*, typically with the help of various *predicates on initially rooted walks* and *predicates on finally rooted walks*. Let $PT(G)$, $PI(G)$ and $PF(G)$, respectively, denote the set of the employed predicates of each of the three kinds. For a walk $\bar{e}$, let $PT(\bar{e})$, $PI(\bar{e})$ and $PF(\bar{e})$ denote the set of those predicates p in $PT(G)$, $PI(G)$ or $PF(G)$, respectively, for which $p(\bar{e})$ is true. We say that the *predicate system* $(PT(G), PI(G), PF(G))$ is a *canonical predicate system* (CPS) if it respects the following rules:

1) For every predicate p in $PT(G)$, there is function θ_p such that for every rooted tour $\bar{e}$,

$$p(\bar{e}) = \exists \bar{e}', e, \bar{e}''.((\bar{e} = \bar{e}' \cdot e \cdot \bar{e}'') \wedge \theta_p(e, PI(\bar{e}'), PF(e \cdot \bar{e}''), PI(\bar{e}' \cdot e), PF(\bar{e}'')))),$$

so that p, if satisfied on $\bar{e}$, has a *witness edge* e, where the evidence for $p(\bar{e})$ is a combination of the properties of e itself, of its past and of its future. An example would be a p supposed to indicate whether a TT $\bar{\tau} = \tau_1, \ldots, \tau_k$ provides sufficient evidence that a

specific transition τ from M is properly implemented. One could define that a τ_i in $\bar{\tau}$ is an adequate witness for $p(\bar{\tau})$ if $lab(\tau_i) = lab(\tau)$, $lab(\tau_1, \ldots, \tau_{i-1})$ recognizes the initial state of τ_i as $init(\tau)$, and $lab(\tau_{i+1}, \ldots, \tau_k)$ recognizes the final state of τ_i as $fin(\tau)$.

2) There is a function β such that for every initially rooted walk $\bar{e} = \bar{e}' \cdot e$, $PI(\bar{e}) = \beta(e, PI(\bar{e}'))$, so that all the predicates in $PI(G)$ can be computed for $\bar{e}$ recursively, by a single forward traversal of $\bar{e}$. In the above example, $PI(\tau_1, \ldots, \tau_{i-1})$ would provide information on the final state of τ_{i-1}, combining the information provided by $lab(\tau_{i-1})$ and by $PI(\tau_1, \ldots, \tau_{i-2})$.

3) There is a function φ such that for every finally rooted walk $\bar{e} = e \cdot \bar{e}'$, $PF(\bar{e}) = \varphi(e, PF(\bar{e}'))$, so that all the predicates in $PF(G)$ can be computed for $\bar{e}$ recursively, by a single backward traversal of $\bar{e}$. In the above example, $PF(\tau_{i+1}, \ldots, \tau_k)$ would provide information on the initial state of τ_{i+1}, combining the information provided by $lab(\tau_{i+1})$ and by $PF(\tau_{i+2}, \ldots, \tau_k)$.

To formally define a CPS, one has to specify its generators $PI(\varepsilon)$, $PF(\varepsilon)$, β, φ and θ_p, that are entirely problem-specific.

2.6. Predicate-Aware Unfolding of Digraphs

Assessment of $PI(\bar{e})$ or $PF(\bar{e})$, respectively, for an initially or a finally rooted walk $\bar{e}$ in a digraph $G = (V, v_0, E)$ with a CPS $(PT(G), PI(G), PF(G))$ typically requires much more than just a brief glance at G. If a tour construction algorithm tends to refer to this information many times, it is reasonable to rewrite G into a form from which the information is directly evident. We define that such a *predicate-aware unfolding* $unf(G)$ of G is a digraph $(V^*(G), E^*(G))$ to which every rooted tour $\bar{e} = e_1, \ldots, e_k$, $e_i = (v_i, v_{i+1}, l_i, \bar{c}_i)$ for $1 \leq i \leq k$, in G

1) for every $1 \leq i \leq (k+1)$ contributes vertex $(v_i, PI(pfx(\bar{e}, i-1)), PF(sfx(\bar{e}, k-i+1)))$, and

2) for every $1 \leq i \leq k$ contributes edge $((v_i, PI(pfx(\bar{e}, i-1)), PF(sfx(\bar{e}, k-i+1))), (v_{i+1}, PI(pfx(\bar{e}, i)), PF(sfx(\bar{e}, k-i))), l_i, \bar{c}_i)$.

Note that there is a one-to-one correspondence between the rooted tours of G and the walks in $unf(G)$ starting in a $(v_0, PI(\varepsilon), PF)$ and ending in a $(v_0, PI, PF(\varepsilon))$. For such a walk $\bar{e}$ in $unf(G)$, the corresponding tour $\bar{e}'$ can be obtained by changing every $((v, PI, PF), (v', PI', PF'), l, \bar{c})$ into $(v, v', l, \bar{c})$, while generation of $\bar{e}$ as $unf(\bar{e}')$ reveals $PI(\bar{e}'')$ or $PF(\bar{e}'')$, respectively, for individual prefixes and suffixes $\bar{e}''$ of $\bar{e}'$.

3. THE CLASSICAL TESTING PROBLEM AND A PETITION FOR A CLEVERER POSTMAN

3.1. The Classical Testing Problem for Finite State Machines

We define CTP as DTGP with the following specializations: 1) M is completely specified and strongly connected. 2) For every two different members of its state set S, there is a sequence of inputs to which the states respond with different sequences of outputs. 3) Every M' has been obtained from M by changing the output and/or the final state of some of its transitions, while not increasing the number of states. 4) One adopts the optimistic assumption on state recognition (OASR) that any state-recognition capability of an I/O sequence $\bar{z}$ from M is preserved in M'. 5) One wants transition-oriented testing, i.e., a TT constructed by combining subsequences testing the implementation of individual transitions in M.

The assumption that M' has no extra states has been included in the CTP definition because it is very common, particularly in RPP-based test generation methods, on which we primarily focus. We are aware that there also exist methods working without the assumption (e.g. [14]), but they typically assume implementation of a reliable reset, which we do not. Let us note that the applicability of our generic method goes beyond the CTP (see, for example, Section 5.8). The very simple setting of the CTP has been introduced primarily to facilitate illustration of how the method relates to similar methods.

A test for a transition $\tau = (s_1, s_2, z, \_)$ corresponds in M to a transition sequence $\bar{\tau} = \bar{\tau}' \cdot \tau \cdot \bar{\tau}'' = (s_3, \_, \bar{z}, \_)$ with $\bar{\tau}' = (s_3, s_1, \bar{z}', \_)$ and $\bar{\tau}'' = (s_2, \_, \bar{z}'', \_)$. When the I/O sequence $\bar{z}$ is observed on an M' from a state supposingly corresponding to s_3, $\bar{z}'$ leads M' to a state s'_1, and then z leads to a state s'_2. The fact that $\bar{z}'$ and $z \cdot \bar{z}''$ have been observed provides some information on s'_1, while the fact that $\bar{z}' \cdot z$ and $\bar{z}''$ have been observed provides some information on s'_2. More precisely, subsequences $\bar{z}'$ and $\bar{z}' \cdot z$ serve for recognizing the state in which they terminate, and subsequences $z \cdot \bar{z}''$ and $\bar{z}''$ for recognizing the state in which they begin.

If an observed I/O sequence $\bar{z}$ distinguishes two states in M, this does not necessarily imply that it distinguishes the corresponding states in M'. Hence, if there is a transition test depending on the capability, one in principle needs additional tests for confirming the capability in M'. As such additional tests might be very expensive, it is desirable to minimize the set of the input sequences employed for state recognition [13, 14]. On the other hand, a non-minimal set might increase opportunities for efficient combining of transition tests, i.e., for their overlapping and for avoiding expensive auxiliary sequences serving exclusively for transferring

M' from the final state of a subtest to the initial state of the next subtest.

Quite short transition-oriented TTs with full discriminating power can be efficiently generated for M with a reliable reset operation [13, 14], but reliable resetting is often too expensive (e.g. time-consuming) to be employed extensively. On the other hand, fully discriminating sequences totally avoiding resets often have a length exponential to the number of states [15]. By adopting OASR, one entirely neglects the need for the additional tests and concentrates on devising tests for individual transitions, such that the cost of the overall test sequence can be minimized. Consequently, the generated TT might fail to have full discriminating power, but testing under OASR is nevertheless of high practical interest, because it can help in the numerous practical cases in which M is so large that minimization of TT cost is of primary importance.

As typical methods for CTP, we expose [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 16]. The methods differ in the additional details of their declared testing strategy, in how precisely the strategy is implemented and/or in the formal models and procedures employed in the implementation. However, they all follow the example of [5] and reduce CTP to an RPP, defined on a graph abstractly representing the structure of M plus some information on the state-recognition power of various I/O sequences and/or on possibilities for subtest overlapping. Still, they differ in the selection of the additional information included in the graph and/or in the form of its representation.

3.2. Non-optimality of the RPP-Based Methods for CTP

GRPP requires that, given a graph and one or more subsets of its edges, one constructs a minimum-cost tour traversing at least one edge from each of the subsets. In a GRPP representing a DTGP, there is one such subset for each of the defined test goals, where each edge in the subset corresponding to a specific test goal represents one or more alternative subtests for satisfying the goal. If one considers a rich choice of alternative subtests, it typically happens that some of the subsets are not singleton, i.e., that the GRPP is not an RPP.

Studying how the existing RPP-based (or other) methods for CTP check the final state of a transition $\tau = (\_, s, \_, \_)$ of a specification FSM M, we observe that those with a *rigid state-recognition scheme* construct such a subtest set $\{\tau \cdot \bar{\tau}_1, \ldots, \tau \cdot \bar{\tau}_k\}$ that $\cup_{i=1\ldots k} FE(lab(\bar{\tau}_i)) = S \setminus \{s\}$, and then require that all the subtests are included in the TT. For every i, let FE_i denote $FE(lab(\bar{\tau}_i))$. In terms of GRPP, the methods for every $1 \leq i \leq k$ try to collect evidence of $fin(\tau) \notin FE_i$, and for every such product g_i establish in E_{g_i} a single provider, an edge representing exactly the subtest $\tau \cdot \bar{\tau}_i$.

Possibilities for TT optimization strongly depend on the exact nature of the state-recognition sequences (SRSs) $\bar{\tau}_i$. A method choosing SRSs in advance, instead of optimally choosing them during TT generation, can hardly find a TT with a minimum cost. The situation is made worse by the fact that such a method usually considers just one, very specific method of state recognition, typically employing a distinguishing sequence (see, for example, [17]), unique input/output (UIO) sequences (see, for example, [5]), or a characterizing set (see, for example, [13]).

To improve chances for finding a minimum-cost TT, the *more flexible methods* for every $\bar{\tau}_i$ provide a set of possible substitutes, typically some $\bar{\tau}'_i$ with $FE(lab(\bar{\tau}'_i)) = FE_i$ (see, for example, methods working with multiple UIO sequences [3, 4]). Only recently, Hierons [1] relaxed the condition to $FE(lab(\bar{\tau}'_i)) \supseteq FE_i$, also observing that the employed SRSs need not be of any special kind, as long as they properly do their job of *state separation*. In terms of GRPP, the only improvement which those methods introduce is that for every g_i, the only member of E_{g_i} becomes capable of satisfying g_i by multiple alternative subtests.

We observe that under OASR, $\{\tau \cdot \bar{\tau}_1, \ldots, \tau \cdot \bar{\tau}_k\}$ can be safely replaced with any $\{\tau \cdot \bar{\tau}'_1, \ldots, \tau \cdot \bar{\tau}'_{k'}\}$ with $\cup_{i=1\ldots k'} FE(lab(\bar{\tau}'_i)) = S \setminus \{s\}$. This is because, in terms of GRPP, the actual products of interest are evidences of $fin(\tau) \neq s'$ for individual s' in $S \setminus \{s\}$, where the subtests satisfying such a $g_{s'}$ are all $\tau \cdot \bar{\tau}'$ in M with $s' \in FE(lab(\bar{\tau}'))$. Obviously, it is not appropriate to give k a special status, i.e., to a priori decide that a specific set of k warehouses must be visited. In other words, optimal implementation of the missing flexibility in state recognition requires that the warehouses to visit are selected during TT construction, implying that one has to switch from RPP to GRPP.

4. REDUCING DTGP TO G^{2}RPP

In this section, we focus entirely on the semantic aspects of reducing DTGP to G^{2}RPP, postponing the discussion of complexity issues and approximations to Section 7. The procedure takes a specification FSM $M = (S, s_0, X, Y, T)$, with its forbidden transitions listed in a T', and a testing strategy pursuing some alternative goal sets, and generates a test $\bar{x}^*$, the input part of the label of a rooted tour in M *satisfying the requirements as precisely as possible and at a minimum cost*. Sections 4.1 to 4.6 describe the consecutive steps of the procedure, whose only part requiring true creativity is formalization of test goals, described in Section 4.2. Section 4.7 proves the optimality of the obtained $\bar{x}^*$. In Section 4.8, we propose a variant of the procedure that accepts a wider class of CPSs, but is less easy to conduct.

4.1. Specification of Additional Cost Criteria and Removal of the Irrelevant Transitions

Let γ^+ denote the cost function γ of M enhanced with the additional cost criteria (if any) of the adopted testing strategy, that also defines the relative importance of the criteria. By enhancing γ into γ^+, M is enhanced into an $M^+ = (S, s_0, T^+)$.

Subsequently, one prunes M^+ into an $M^* = (S^*, s_0, T^*)$, the maximal strongly connected and in s_0 rooted subgraph of the graph $(S, s_0, \{(s, s', x/y, \gamma^+(s, x))| (s, s', x/y, \gamma(s, x)) \in T \setminus T'\})$, removing all transitions which are forbidden or cannot lie on a rooted tour. M^* is the FSM on which $\bar{x}^*$ is subsequently constructed as $in(\bar{\tau}^*)$ of an optimal rooted tour $\bar{\tau}^*$.

4.2. Constructing a CPS

To formalize the goals of the adopted testing strategy, one constructs for M^* a CPS $(PT(M^*), PI(M^*), PF(M^*))$, i.e., defines its generators $PI(\varepsilon)$, $PF(\varepsilon)$, β, φ and θ_p. The predicates in $PT(M^*)$ are the *predicates of interest* on $\bar{\tau}^*$. The only motivation for including predicates into $PI(M^*)$ or $PF(M^*)$ is to make $(PT(M^*), PI(M^*), PF(M^*))$ a CPS.

The CPS is constructed as a *complete set of instructions on how the predicates of interest can be evaluated* on rooted tours in M^*. In other words, the constructed CPS is some kind of a tour-evaluation program written in an entirely *declarative style* (remember logic programming languages, e.g. Prolog [18]).

To simplify test generation, we in the *basic variant* of the procedure require that the CPS respects some additional restrictions:

RESTRICTION 1. $PI(\varepsilon)$ must be empty.

RESTRICTION 2. β must be such that for every transition τ in T^* and $PI \subseteq PI' \subseteq PI(M^*)$, $\beta(\tau, PI) \subseteq \beta(\tau, PI')$.

RESTRICTION 3. $PF(\varepsilon)$ must be empty.

RESTRICTION 4. φ must be such that for every transition τ in T^* and $PF \subseteq PF' \subseteq PF(M^*)$, $\varphi(\tau, PF) \subseteq \varphi(\tau, PF')$.

RESTRICTION 5. Every θ_p must be such that for every transition τ in T^*, $PI_1 \subseteq PI_1' \subseteq PI(M^*)$, $PF_1 \subseteq PF_1' \subseteq PF(M^*)$, $PI_2 \subseteq PI_2' \subseteq PI(M^*)$ and $PF_2 \subseteq PF_2' \subseteq PF(M^*)$, $\theta_p(\tau, PI_1, PF_1, PI_2, PF_2) \Rightarrow \theta_p(\tau, PI_1', PF_1', PI_2', PF_2')$.

CPS construction starts by choosing the constituents of $PT(M^*)$ and setting $PI(M^*)$ and $PF(M^*)$ to empty. One then refines predicates in $PT(M^*)$ and, where necessary, also predicates in $PI(M^*)$ and $PF(M^*)$, if any introduced, into functions of more and more elementary predicates on initially or finally rooted walks, where any additionally introduced auxiliary predicate is included into $PI(M^*)$ or $PF(M^*)$, as appropriate. Typically, predicates in $PT(M^*)$ are not very complicated, so that the refinement is not unreasonably difficult to conduct (see Section 5.1 for an example). CPS construction terminates when every non-elementary predicate in $PT(M^*)$, $PI(M^*)$ or $PF(M^*)$ has been adequately expressed in terms of predicates already in $PI(M^*)$ or $PF(M^*)$, so that no additional auxiliary predicates are needed.

It is advisable that for every predicate p in $PT(M^*)$, $PT(M^*)$ also comprises some or all of those *abstractions* of p which can be encoded without further extending $PI(M^*)$ or $PF(M^*)$. An abstraction of a predicate p is an implication of it, i.e., an approximation of p which can be satisfied more easily than p. The suggestion is motivated by the fact that in the case that for a predicate p in $PT(M^*)$, no satisfactory tour exists, one becomes interested in tours satisfying at least its selected abstractions (see Section 4.4). On the other hand, if a tour satisfying p is found, all its abstractions are satisfied automatically. For example, for a predicate p requiring for a transition $\tau = (\_, s, \_, \_)$ a subtest $\tau \cdot \bar{\tau}$ with $FE(lab(\bar{\tau})) = S \setminus \{s\}$, i.e., a test checking the final state s of τ with an UIO sequence, it is advisable that $PT(M^*)$ for every s' in $S \setminus \{s\}$ also comprises the abstraction of p requiring for $\bar{\tau}$ just $s' \in FE(lab(\bar{\tau}))$, because construction of an SRS separating s from multiple states simultaneously might be impossible, so that one becomes interested in subtests separating s from individual states.

In terms of G^{2}RPP, each predicate p in $PT(M^*)$ is a *goal*, i.e., a *product of interest*, while its availability is at this stage of TT construction not yet known. One nevertheless defines on $PT(M^*)$ a *partition* Ξ^+, taking care that if a goal set (i.e., a product set) ξ in Ξ^+ comprises a predicate p, it also comprises every predicate in $PT(M^*)$ which is an abstraction of p.

4.3. Unfolding the FSM

To represent the rooted tours of M^* in a form explicitly showing to which extent their prefixes and suffixes satisfy the predicates in the CPS, one constructs the unfolding $unf(M^*) = G^* = (V^*, E^*)$. An algorithm is given in Figure 1. In its first phase, one traverses every rooted tour in the forward direction, to compute and encode the predicates on its prefixes. In the second phase, one traverses every rooted tour in the backward direction, to compute and encode the predicates on its suffixes.

In comparison with the CPS, the algorithm is a *procedural representation of tour evaluation*. The generated $unf(M^*)$, if interpreted as in Section 4.4, is a *concise representation of evaluation results for every single rooted tour* in M^*, and as such an excellent basis for searching for an optimal tour.

$Open := \{(s_0, PI(\varepsilon))\}; V' := \emptyset; E' := \emptyset;$
while $Open \neq \emptyset$
  Move an (s, PI) from $Open$ to V';
  $Edges := \{((s, PI), (s', \beta((s, s', x/y, \bar{c}), PI)), x/y, \bar{c})$
    $|(s, s', x/y, \bar{c}) \in T^*\}$;
  $Open := Open \cup (\{v'|\exists(\_, v', \_, \_) \in Edges\} \setminus V')$;
  $E' := E' \cup Edges$ **endwhile**;
$Open := \{(s_0, PI, PF(\varepsilon))|(s_0, PI) \in V'\}$;
$V^* := \emptyset; E^* := \emptyset;$
while $Open \neq \emptyset$
  Move an (s', PI', PF') from $Open$ to V^*;
  $Edges := \{((s, PI, \varphi((s, s', x/y, \bar{c}), PF')),$
    $(s', PI', PF'), x/y, \bar{c})$
    $|((s, PI), (s', PI'), x/y, \bar{c}) \in E'\}$;
  $Open := Open \cup (\{v|\exists(v, \_, \_, \_) \in Edges\} \setminus V^*)$;
  $E^* := E^* \cup Edges$ **endwhile**

FIGURE 1. Computation of $unf(M^*)$.

4.4. Assessment of Warehouses

The very specific nature of the predicates in $PT(M^*)$ implies that such a predicate p is *satisfied on a rooted tour* $\bar{\tau}$ in M^*, i.e., that p is in $PT(\bar{\tau})$, exactly if in the tour unfolding $unf(\bar{\tau}) = e_1, \ldots, e_k$, which is a walk in G^*, an edge e_i with $1 \leq i \leq k$ acts as a *witness* for $p(\bar{\tau})$, i.e., in terms of G^{2}RPP, if the warehouse v_{e_i} hosted by the edge stocks the product p. For an edge $e = ((s, PI, PF), (s', PI', PF'), l, \bar{c})$ in E^*, the *stock* $PT(e)$ of v_e consists of all products p with $\theta_p((s, s', l, \bar{c}), PI, PF, PI', PF')$ true.

The satisfiable goals are those in a $PT^+ = \cup_{e \in E^*} PT(e)$. Knowing PT^+, one prunes Ξ^+ into a Ξ^*, the *closest approximation comprising exclusively fully satisfiable goal sets*, as follows: If Ξ^+ comprises a goal set $\xi \subseteq PT^+$, i.e., a ξ satisfiable in its entirety, Ξ^* is the set of all such ξ in Ξ^+. If no such goal set exists, one becomes interested in satisfying at least an approximation of a goal set in Ξ^+, defining that Ξ^* for every ξ in Ξ^+ comprises its largest satisfiable subset $\xi \cap PT^+$. The *new set of the products of interest* is a $PT^* = \cup_{\xi \in \Xi^*} \xi$. A TT $\bar{\tau}^*$ is considered *optimal* if there is in M^* no rooted tour satisfying a ξ in Ξ^* at a lower cost.

Note that at this point, those predicates in $PI(M^*)$ and $PF(M^*)$ which serve exclusively for evaluation of predicates in $PT(M^*) \setminus PT^*$, become obsolete. Hence, one might want to *re-compute* $unf(M^*)$ (and the stock of its warehouses) without them, to get a smaller G^*.

4.5. Completing the G^{2}RPP Specification

G^* is enhanced into a graph $G = (V, v_0, E)$ that is (as G^* might not be) rooted, strongly connected and with a loop in the root, and can, hence, be fed to a G^{2}RPP solver for production of a rooted tour. Completing G^* into G, one introduces v_0 as an auxiliary vertex, and the following auxiliary edges:

1) a *root loop* $e_0 = (v_0, v_0, \varepsilon, \bar{0})$, so that it can be required that the constructed tour is rooted (see the last paragraph of Section 2.4),

2) a *starting edge* $(v_0, (s_0, PI(\varepsilon), PF), \varepsilon, \bar{0})$ for every vertex $(s_0, PI(\varepsilon), PF)$ in V^*, and

3) an *ending edge* $((s_0, PI, PF(\varepsilon)), v_0, \varepsilon, \bar{0})$ for every vertex $(s_0, PI, PF(\varepsilon))$ in V^*.

The goal set (i.e. the product set) Γ for the G^{2}RPP is $PT^* \cup \{g_0\}$, where the goal g_0 denotes traversal of e_0. Hence, $E_{g_0} = \{e_0\}$, while for the other products in Γ, the warehouses providing them are those residing in the subgraph G^*. Ξ for the G^{2}RPP is constructed from Ξ^* by enhancing each of its members with g_0.

4.6. Interpreting the Constructed Tour

As G is strongly connected, it has a rooted tour traversing every edge, i.e., visiting every warehouse in G^* and thereby collecting every product p in PT^*, implying that the constructed G^{2}RPP indeed has a solution, a rooted tour $\bar{e}^*$ in G which is optimal with respect to Ξ^*. Deleting its elements corresponding to auxiliary edges, one obtains such a sequence $\bar{e}^+ = e_1, \ldots, e_k$, $e_i = ((s_i, PI_i, PF_i), (s_{i+1}, PI'_{i+1}, PF'_{i+1}), l_i, \bar{c}_i)$ for $1 \leq i \leq k$, that $\bar{\tau}^* = \tau_1, \ldots, \tau_k$, $\tau_i = (s_i, s_{i+1}, l_i, \bar{c}_i)$ for $1 \leq i \leq k$, is in M^* a rooted tour, and $in(\bar{\tau}^*)$ is one of the optimal tests for M and T' under the adopted testing strategy. As proved in Section 4.7, this is true even if $\bar{e}^+$ is not exactly $unf(\bar{\tau}^*)$.

4.7. The Optimality of the Obtained Test

Solving the derived G^{2}RPP, one obtains a rooted tour $\bar{e}^*$ which might pass through v_0 several times. $\bar{e}^+$, the projection of $\bar{e}^*$ onto the edges in E^*, is, hence, an $\bar{e}_1 \cdot \ldots \cdot \bar{e}_n$ where every segment $\bar{e}_i = e_{i,1}, \ldots, e_{i,k_i}$, $e_{i,j} = ((s_{i,j}, PI_{i,j}, PF_{i,j}), (s_{i,j+1}, PI_{i,j+1}, PF_{i,j+1}), l_{i,j}, \bar{c}_{i,j})$ for $1 \leq i \leq n$ and $1 \leq j \leq k_i$, is $unf(\bar{\tau}_i)$ of some rooted tour $\bar{\tau}_i$ in M^*, and $\bar{\tau}^* = \bar{\tau}_1 \cdot \ldots \cdot \bar{\tau}_n$ is a rooted tour in M^* representing the final result of the TT construction procedure, where $unf(\bar{\tau}^*)$ is an $e_1, \ldots, e_k$, $e_m = ((s_m, PI_m, PF_m), (s_{m+1}, PI_{m+1}, PF_{m+1}), l_m, \bar{c}_m)$ for $1 \leq m \leq k$.

$unf(\bar{\tau}^*)$ has the property that for every $1 \leq m \leq k$ and the corresponding e_m in $unf(\bar{\tau}^*)$ and $e_{i,j}$ in $\bar{e}^+$, $PI_{i,j} \subseteq PI_m$, $PI_{i,j+1} \subseteq PI_{m+1}$, $PF_{i,j} \subseteq PF_m$ and $PF_{i,j+1} \subseteq PF_{m+1}$. To prove the first two relations, observe that to recursively compute PI_m and PI_{m+1}, one computes the corresponding attributes for the edges preceding e_m, traversing $\bar{\tau}^*$ in the forward direction. To obtain $PI_{i,j}$ and $PI_{i,j+1}$, one follows basically the same procedure, except that for every e_{q,k_q} in $\bar{e}^+$, corresponding to some $e_{m'}$ in $unf(\bar{\tau}^*)$, $PI_{q+1,1}$ is not simply a copy of PI_{q,k_q+1}, but rather set to $PI(\varepsilon)$. By Restriction 1, $PI(pfx(\tau^*, m'))$ can only be *under*estimated by the modification, and, by Restriction 2, this can only lead to *under*estimation of $PI(pfx(\tau^*, m''))$ for $m' < m'' \leq m' + k_{q+1}$. An analogous reasoning can be employed for proving that

$PF(sfx(\tau^*, m'))$ for $0 \leq m' \leq k$ is never overestimated in $\bar{e}^+$, except that one now traverses $\bar{\tau}^*$ in the backward direction and relies on Restrictions 3 and 4.

The above proven property of $unf(\bar{\tau}^*)$ by Restriction 5 implies that whenever $\bar{e}^*$ traverses a $cmp(\bar{e}^+, m)$ for collecting a product p, p is indeed in $PT(cmp(unf(\bar{\tau}^*), m))$, because the stock of a warehouse is never overestimated. Hence, if $\bar{e}^*$ collects all products in a product set ξ in Ξ^*, so does $unf(\bar{\tau}^*)$, i.e., $\bar{\tau}^*$ satisfies every goal in ξ.

Now suppose that M^* had a rooted tour $\bar{\tau}$ satisfying a sufficient set of goals at a cost less than $cost(\bar{\tau}^*)$ (i.e., $cost(\bar{e}^*)$). Then G would have a rooted tour starting with the initial loop, following an auxiliary edge towards the start of $unf(\bar{\tau})$, traversing $unf(\bar{\tau})$ and returning to v_0 over an auxiliary edge. The tour would collect a sufficient set of products and have the cost of $\bar{\tau}$, contradicting the fact that $\bar{e}^*$ is one of the minimum-cost rooted tours in G collecting a sufficient set of products. Hence, such $\bar{\tau}$ cannot exist and we conclude that $\bar{\tau}^*$ is one of the TTs which are optimal for the given T', additional cost criteria, CPS and Ξ^+, implying that $in(\bar{\tau}^*)$ is among the optimal tests.

4.8. A Variant of the Procedure

Restrictions 1 to 5 can be inconvenient (see, for example, Section 5.8). Fortunately, they are, as evident from Section 4.7, relevant only if $\bar{e}^*$ passes from v_0 to G^* more than once. Hence, one may as well drop them, provided that the procedure is modified as follows:

For G, one introduces an additional cost criterion of maximal importance. With respect to the criterion, all edges of the subgraph G^* are of cost 0, and all the auxiliary edges of cost 1, so that the G^{2}RPP solver minimizes traversal of v_0. If $\bar{e}^*$ still passes from v_0 to G^* more than once, this is an indication that there is no $\bar{\tau}^*$ satisfying Ξ^*, implying that one should resort to its approximations.

The inconvenience of the modified procedure is that one might have to try several approximations of Ξ^* before finding a satisfiable one. Besides, the modified procedure is much less easy to conduct in an approximate way (see Section 7).

5. SOME STRATEGIES FOR DTGP

In Section 4.7, we *proved* that for any testing strategy it accepts, our method generates a test satisfying the goals of the strategy as much as the non-forbidden transitions of M allow and at a minimum cost. Hence, *the question whether, for a specific testing strategy, the method generates an optimal test, reduces to the question whether the strategy can be formalized in the required way.*

In this section, we discuss the problem for some typical strategies for DTGP. For the simpler ones, we provide a precise formalization of the required form, thereby proving that our method is their exact implementation. For each of the proposed $PT(M^*)$, we describe only those of its members which are not abstractions of the others, *assuming that any abstraction of the described predicates is*, following the advice from Section 4.2, *included* into $PT(M^*)$ *by default.*

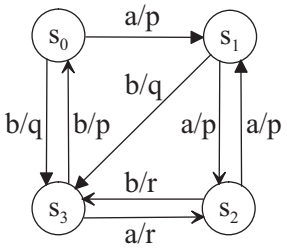

FIGURE 2. A deterministic FSM.

5.1. A Strategy for CTP Checking Only the Final State of Transitions

Remember that for CTP, we recommend reduction to GRPP, i.e., $\Xi^+ = \{PT(M^*)\}$, $\Xi^* = \{PT^+\}$ and $PT^* = PT^+$, implying that TT $\bar{\tau}^*$ satisfies all the satisfiable goals. For an M with $|S| > 1$, if one is interested primarily in the *minimization of test cost*, then one typically requires that for every transition $\tau = (s, s', z, \_)$ in T^* and state s'' in $S \setminus \{s'\}$, $\bar{\tau}^*$ comprises a $\tau \cdot \bar{\tau}_{\tau,s''}$ with $s'' \in FE(lab(\bar{\tau}_{\tau,s''}))$, not really caring about the exact nature of the SRSs $\bar{\tau}_{\tau,s''}$ employed for checking the final state of τ [1, 2, 3, 4, 5, 6, 7, 9, 10].

In a CPS formalizing such a strategy, $PI(M^*)$ can be empty, while in $PF(M^*)$, we need for every state s in S a predicate $p'_s(\bar{\tau})$ equivalent to $(s \in FE(lab(\bar{\tau})))$. A well-formed definition for such a $p'_s(\bar{\tau})$ would be

$$\exists \tau', \bar{\tau}'.((\bar{\tau} = \tau' \cdot \bar{\tau}') \wedge \forall s' \in S.(\neg p'_{s'}(\bar{\tau}') \Rightarrow \neg \exists (s, s', lab(\tau'), \_) \in T))$$

$PT(M^*)$ would then for every transition $\tau = (s, s', z, \_)$ in T^* and state $s'' \in S \setminus \{s'\}$ comprise a predicate $p_{\tau,s''}(\bar{\tau})$ defined as

$$\exists \bar{\tau}', \tau', \bar{\tau}''.((\bar{\tau} = \bar{\tau}' \cdot \tau' \cdot \bar{\tau}'') \wedge (init(\tau') = s) \wedge (lab(\tau') = z) \wedge p'_{s''}(\bar{\tau}''))$$

Among the earlier methods following the strategy, the most advanced one is that of [1], excelling in a most flexible state-recognition scheme and also supporting subtest overlapping. Therefore, let us for illustration take the same M as Hierons and see if we can really generate an even shorter test. The FSM is depicted in Figure 2. Like [1], we assume that all transitions are non-forbidden and of the same cost.

The constructed G is given in Figure 3, in which every vertex $(s_i, \{\}, PF)$ is encoded as $s_i^{\{j|p'_{s_j} \in PF\}}$. In the G, there are providers for every specified $p_{\tau,s''}$, implying that $\Gamma = PT(M^*) \cup \{g_0\}$. For example, adequate witnesses for $p_{(s_0,s_1,a/p,\_),s_3}$ are the edges $(s_0^{\{1,3\}}, s_1^{\{2,3\}}, a/p, \_)$ and $(s_0^{\{1,3\}}, s_1^{\{0,2,3\}}, a/p, \_)$. Still,

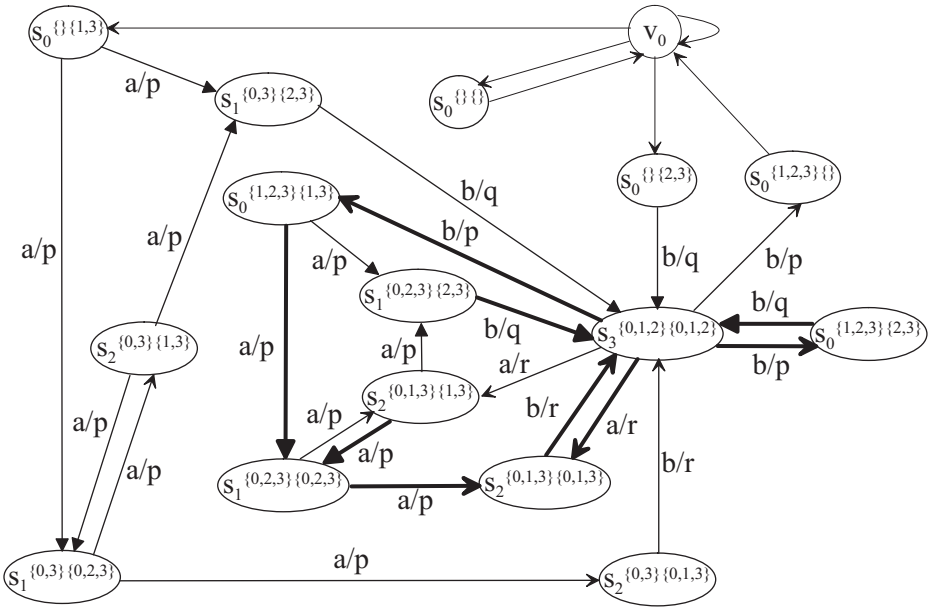

FIGURE 4. The graph derived for the FSM in Figure 2 under the strategy described in Section 5.2.

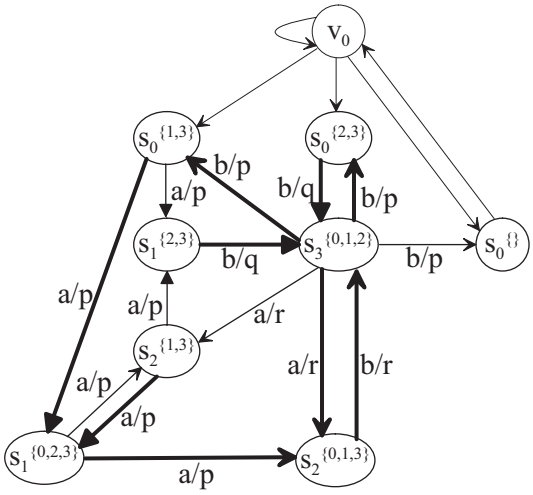

FIGURE 3. The graph derived for the FSM in Figure 2 under the strategy described in Section 5.1.

the overall distribution of products in the warehouses of G is for the particular M such that this GRPP is an RPP, where the edges requiring traversal are those drawn bold. A possible TT would be

$$s_0 \xrightarrow{a/p} s_1 \xrightarrow{a/p} s_2 \xrightarrow{a/p} s_1 \xrightarrow{a/p} s_2 \xrightarrow{b/r} s_3 \xrightarrow{a/r} s_2 \xrightarrow{b/r} s_3 \xrightarrow{b/p} s_0 \xrightarrow{a/p} s_1 \xrightarrow{b/q} s_3 \xrightarrow{b/p} s_0 \xrightarrow{b/q} s_3 \xrightarrow{b/p} s_0$$

comprising only 13 steps. The TT is 2 steps *shorter* than the one which Hierons, pursuing the same test goals, produced in [1]. In the particular case, the difference is caused by Hierons's slight bias towards an initially selected set of SRSs. In our method, the problem does not exist, because there is *no explicit SRS construction*!

Now suppose that $PT(M^*)$ is further simplified by deletion of $p_{(s_0,s_1,a/p,\_),s_0}$, for example, because it is known that no M' in Φ_M responds to a in s_0 by a transition terminating in s_0. With the simplification, it is no longer appropriate to a priori decide to obtain $p_{(s_0,s_1,a/p,\_),s_3}$ from the second one of its providers, i.e., from the only witness for the now abandoned predicate, because one may as well decide not to visit the provider. With the additional freedom of choice, the GRPP is no longer an RPP.

5.2. A Strategy for CTP Checking Also the Initial State of Transitions

To increase the discriminating power of the generated test, the strategy described in Section 5.1 is often enhanced by requiring that in every transition test, the initial state of the transition is also checked. Typically, the SRS employed for the purpose is a BUIO (backward UIO) sequence, i.e., an UIO sequence extending in the past of the state [16].

In a CPS formalizing the strategy, $PF(M^*)$ can be as in Section 5.1, while in $PI(M^*)$, we need for every state s in S a predicate $p_s(\bar{\tau})$ equivalent to $(s \in BE(lab(\bar{\tau})))$. A well-formed definition of such a $p_s(\bar{\tau})$ would be

$$\exists \bar{\tau}', \tau'.((\bar{\tau} = \bar{\tau}' \cdot \tau') \wedge \forall s' \in S.(\neg p_{s'}(\bar{\tau}') \Rightarrow \neg \exists (s', s, lab(\tau'), \_) \in T))$$

$PT(M^*)$ would then for every transition $\tau = (s, s', z, \_)$ in T^* and state $s'' \in S \setminus \{s'\}$ comprise a predicate $p_{\tau,s''}(\bar{\tau})$ defined as

$$\exists \bar{\tau}', \tau', \bar{\tau}''.((\bar{\tau} = \bar{\tau}' \cdot \tau' \cdot \bar{\tau}'') \wedge (\forall s''' \in S \setminus \{s\}.p_{s'''}(\bar{\tau}')) \wedge (init(\tau') = s) \wedge (lab(\tau') = z) \wedge p'_{s''}(\bar{\tau}''))$$

For the example M in Fig. 2, the enhanced G is given in Figure 4, in which every vertex (s_i, PI, PF)

is encoded as $s_i^{\{j|p_{s_j}\in PI\}\{j|p'_{s_j}\in PF\}}$. The overall distribution of products in the warehouses of G is again such that this GRPP is an RPP, where the edges requiring traversal are those drawn bold. A possible TT would be

$$s_0 \overset{b/q}{\rightarrow} s_3 \overset{b/p}{\rightarrow} s_0 \overset{b/q}{\rightarrow} s_3 \overset{b/p}{\rightarrow} s_0 \overset{a/p}{\rightarrow} s_1 \overset{a/p}{\rightarrow} s_2 \overset{a/p}{\rightarrow}$$
$$s_1 \overset{b/q}{\rightarrow} s_3 \overset{a/r}{\rightarrow} s_2 \overset{a/p}{\rightarrow} s_1 \overset{a/p}{\rightarrow} s_2 \overset{b/r}{\rightarrow} s_3 \overset{a/r}{\rightarrow} s_2 \overset{b/r}{\rightarrow} s_3 \overset{b/p}{\rightarrow} s_0$$

of length 15. This is exactly the length of the TT constructed in [1], but our TT *implements more ambitious test goals*. For example, the TT from [1] checks transition $(s_1, s_2, a/p, \_)$ without checking its initial state.

5.3. A Strategy for CTP Equally Trusting Backward and Forward SRSs

Trying to minimize the test cost, one might incline to equally trust backward SRSs, i.e., those extending in the past of the state being checked, and forward SRSs, i.e., those extending in the future of the state [8].

Modifying in this direction the strategy described in Section 5.2, one can leave $PI(M^*)$ and $PF(M^*)$ as they are, implying that it is not necessary to re-compute G. In $PT(M^*)$, it suffices to redefine each $p_{\tau,s''}(\bar{\tau})$ as

$$\begin{aligned}\exists\bar{\tau}', \tau', \bar{\tau}''.((\bar{\tau} = \bar{\tau}' \cdot \tau' \cdot \bar{\tau}'')\wedge\\ (\forall s''' \in S \setminus \{s\}.(p_{s'''}(\bar{\tau}') \vee p'_{s'''}(\tau' \cdot \bar{\tau}'')))\wedge\\ (init(\tau') = s) \wedge (lab(\tau') = z)\wedge\\ (p_{s''}(\bar{\tau}' \cdot \tau') \vee p'_{s''}(\bar{\tau}')))\end{aligned}$$

thereby usually enhancing the set of its providers in G, i.e., increasing the freedom of subtest choice.

As demonstrated in [8], such strategy modification can lead to substantially shorter tests. If the modified strategy is implemented precisely, the benefit can become even more evident.

5.4. A Strategy for CTP With Double-Checking of States

A scrupulous tester, being aware that OASR is naive, would want to check each state of interest both as the final state of a past and as the initial state of a future, using SRSs believed to be as strong as possible. In this section, we modify in this direction the strategy described in Section 5.2, to the maximum extent possible without enhancing $PI(M^*)$ or $PF(M^*)$.

For such a strategy, $PT(M^*)$ would for every transition $\tau = (s, s', z, \_)$ in T^* comprise a predicate $p_\tau(\bar{\tau})$ defined as

$$\begin{aligned}\exists\bar{\tau}', \tau', \bar{\tau}''.((\bar{\tau} = \bar{\tau}' \cdot \tau' \cdot \bar{\tau}'')\wedge\\ (\forall s'' \in S \setminus \{s\}.(p_{s''}(\bar{\tau}') \wedge p'_{s''}(\tau' \cdot \bar{\tau}'')))\wedge\\ (init(\tau') = s) \wedge (lab(\tau') = z)\wedge\\ (\forall s'' \in S \setminus \{s'\}.(p_{s''}(\bar{\tau}' \cdot \tau') \wedge p'_{s''}(\bar{\tau}''))))\end{aligned}$$

Considering also the predicate abstractions present in $PT(M^*)$ by default, this $PT(M^*)$ is a superset of those defined in Sections 5.1, 5.2 and 5.3. The previously defined strategies are, hence, just typical simplifications of the one just defined.

The scrupulous strategy has the interesting property that the resulting GRPP is *always an RPP* (see Section 6.1 for a proof). However, as demonstrated by the example at the end of Section 5.1, not all its simplifications inherit the property.

For illustration, let us again consider the M in Figure 2 and its G in Figure 4. For the just defined strategy, $PT(M^*)$ for the example FSM comprises some predicates for which no provider exists in G. Such is, for example, $p_{(s_3,s_0,b/p,\_)}$, because s_0 possesses no UIO sequence. Detecting its non-satisfiability, the method *automatically resorts to its abstractions* with the forward SRS in the final state of the transition $(s_3, s_0, b/p, \_)$ distinguishing s_0 just from the states in $\{s_1, s_3\}$ or $\{s_2, s_3\}$, respectively.

Among the predicates which are satisfiable and not an abstraction of another satisfiable predicate, each has in G a single provider edge. In Figure 4, the edges are exactly those drawn bold, i.e., requiring traversal already by the strategy described in Section 5.2, implying that for the particular M, the two strategies are equivalent.

5.5. Employing Robust SRSs

A cautious tester does not assume that, for example, any I/O sequence $\bar{z}$ in an M with $FE(\bar{z}) = S \setminus \{s\}$ can in an M' reliably recognize any state s' corresponding to s, but requires that $\bar{z}$ also possesses some specific properties, e.g., that it checks, with its respective suffixes, not only s', but also some of the subsequent states [19]. To implement such precautions with our method, it suffices to make functions θ_p depend on additional predicates in $PF(M^*)$ telling whether the label of a given $\bar{\tau}$ possesses the properties additionally required for an SRS. To secure additional properties for the employed backward SRSs, one would enhance $PI(M^*)$. Selection of robust SRSs thereby becomes an integral part of the TT optimization process.

5.6. Pursuing Avoidance of Critical or Less Trusted Transitions

To increase the discriminating power of the generated test, one might want to employ the two additional transition cost criteria described below, with their relative importance tuned to the specific situation.

Anido and Cavalli [20] demonstrated that a TT produced by combining tests of individual transitions might fail to detect that a given M' is faulty even if pre-tests have confirmed that the sequences employed for state recognition have the expected power. They also suggested that, to avoid the situation, TT should, as much as possible, avoid *critical converging transitions*. A transition $\tau = (s, s', z, \bar{c})$ in T is converging if it is not the only member of T of the form $(\_, s', z, \_)$. Such a τ is critical if there is in Φ_M an M' with a faulty transition of this form. Maximum avoidance of critical

converging transitions can be implemented simply by declaring them more expensive.

Another interesting additional cost criterion for an individual transition $(s, s', x/y, \_)$ would be the degree to which its implementation is trusted to be correct. It certainly should be trusted if in M and in every M' in Φ_M, the input x in every state results in the output y and the state s'. Typical such trustworthy transitions are reliable resets, if implemented. Like critical converging transitions, *less trusted transitions* are undesirable in auxiliary roles, i.e., as constituents of sequences serving exclusively for state recognition and/or for transfer. One would, hence, want to minimize inclusion of less trusted transitions, by declaring them more expensive.

Test goal satisfaction and avoidance of undesirable transitions are two conflicting optimization issues. Although more ambitious test goals in principle increase the discriminating power of the generated TT, they often require a TT with a larger number of critical converging or less trustworthy transitions, implying that the obtained TT might be less trustworthy than expected.

In the example M in Figure 2, the converging transitions are, like their copies in Figures 3 and 4, marked with a triangle arrowhead. It turns out that the TT given for M in Section 5.2 and nearly perfectly satisfying even the ambitious goals given in Section 5.4, i.e., more ambitious goals than the TT of the same length constructed in [1], comprises more traversals of converging transitions than the TT.

5.7. Transition-Oriented Construction of Checking Tours

A checking tour (CT) for an M is a TT failing on every M' in Φ_M. In transition-oriented construction of CTs, the focus of interest is not on subtests for individual transitions, but on the *cumulative benefit of subtests* individually serving for verification of some transitions and/or for verification of some of the SRSs employed in transition verification. In other words, a flexible transition-oriented checking strategy aims at a CT satisfying *one of many satisfactory sets* of predicates, i.e., requires that a DTGP is reduced to a G^{2}RPP.

Among the existing methods for transition-oriented construction of CTs, e.g. [15, 21, 22, 23, 24, 25], the more recent ones are quite flexible, but still pursue rather rigid state-recognition schemes and fail to handle the various optimization concerns in a truly integrated manner. With our method, one could implement more flexible schemes, potentially leading to cheaper CTs.

5.8. Fault-Model-Driven Test Construction

For an M, a CT constructed by a method precisely implementing a specific checking strategy is a minimum-cost CT among the CTs implementing the strategy, but not necessarily among *all* CTs for the given Φ_M. As Anido and Cavalli wonder in [20], one occasionally encounters a CT for which none of the known checking strategies seems to explain how it can be so powerful in spite of being so cheap.

Our answer is that, to generate an *absolutely optimal CT*, one needs a strategy pursuing *exactly the absolutely necessary test goals*, i.e., a strategy accepting *any* non-forbidden input sequence producing a wrong response for every M' in Φ_M, while returning M to its root. The existing methods for CT construction fail to meet the requirement because they from the beginning pursue just a specific kind of sequences (i.e., over-ambitious test goals), thereby potentially overlooking the optimal ones.

With our method, actually with its variant from Section 4.8, the optimal (i.e., minimum-requirements) strategy can be implemented by the following CPS neglecting Restrictions 1 and 2: $PT(M^*)$ for every $M' = (S_{M'}, s_{M'}, T_{M'})$ in Φ_M comprises a predicate $p_{M'}(\bar{\tau})$ defined as

$\exists \bar{\tau}', \tau, \bar{\tau}'', s \in S_{M'}.$
$\quad((\bar{\tau} = \bar{\tau}' \cdot \tau \cdot \bar{\tau}'') \wedge p_{M',s}(\bar{\tau}') \wedge$
$\quad\exists(s, \_, in(\tau)/y, \_) \in T_{M'}.(y \neq out(\tau)))$

where a $p_{M',s}(\bar{\tau})$ in $PI(M^*)$ is defined as

$((\bar{\tau} = \varepsilon) \wedge (s = s_{M'})) \vee$
$\exists \bar{\tau}', \tau, s' \in S_{M'}.((\bar{\tau} = \bar{\tau}' \cdot \tau) \wedge p_{M',s'}(\bar{\tau}') \wedge$
$\qquad\exists(s', s, in(\tau)/\_, \_) \in T_{M'})$

An M possesses a CT for a Φ_M and a T' exactly if an $\bar{e}^*$ passing from v_0 to G^* only once and satisfying the entire $PT(M^*)$ can be constructed (see Section 4.8). In that case, there is for an individual M' in Φ_M often more than one way in which it can prove incompatible with M, i.e., in the derived GRPP, a product might be available in more than one warehouse. Likewise, an individual step of a test often proves incorrectness of more than one M' in Φ_M, i.e., in the derived GRPP, a warehouse might be stocking more than one product. Hence, construction of an optimal CT in a general case reduces to a GRPP which is not an RPP. Thereby, we have proven that *the nature of DTGP is that of GRPP.*

6. SOLVING G^{2}RPP

6.1. Deleting Redundant Elements of a G^{2}RPP Definition

A G^{2}RPP defined as in Section 2.4 might allow further simplifications. For every goal set ξ in Ξ, let $\Upsilon(\xi)$ denote the set of all minimal subsets η of $\cup_{g \in \xi} E_g$ such that for every goal g in ξ, $\eta \cap E_g \neq \emptyset$. If for a goal set ξ in Ξ, there is in Ξ a $\xi' \neq \xi$ with $\Upsilon(\xi) \subseteq \Upsilon(\xi')$, ξ may be deleted from Ξ, because every tour satisfying ξ is among the tours satisfying ξ'.

For a goal g in a goal set ξ in Ξ, let $\Upsilon_g(\xi)$ denote the set of all minimal subsets η of $\cup_{g' \in (\xi \setminus \{g\})} E_{g'}$ such that for every g' in $\xi \setminus \{g\}$, $\eta \cap E_{g'} \neq \emptyset$. If for every η in $\Upsilon_g(\xi)$, $\eta \cap E_g \neq \emptyset$, g may be deleted from ξ, because every tour satisfying $\xi \setminus \{g\}$ also satisfies g.

After deleting all the redundant product sets and product set members, one re-computes Γ as $\cup_{\xi\in\Xi}\xi$. Then for every goal g in Γ, one deletes from E_g every edge e not in $\cup_{\eta\in\Upsilon(\xi),\xi\in\Xi}\eta$, thereby deleting g from the stock of the warehouse v_e, because in every tour satisfying Ξ by satisfying a goal set ξ comprising g, there are providers of g other than v_e. By deleting all the redundant stocks, one virtually deletes every v_e with e no longer in $\cup_{g\in\Gamma}E_g$.

It might turn out that the simplified G^{2}RPP is actually a GRPP or even an RPP, and can be solved as such. Take, for example, the GRPP constructed for a CTP under the strategy described in Section 5.4. For every predicate p representing a goal in Γ, if there are in E_p two different edges e and e', PT^* also comprises a predicate p' with $E_{p'}$ comprising just one of the edges, i.e., a p' of which p is an abstraction, implying that p is redundant in Γ. Hence, no goal p with E_p non-singleton can survive the simplification of Γ. As E_{g_0} is also singleton, the GRPP is an RPP.

6.2. Reducing G^{2}RPP to G^{2}TSP

A G^{2}RPP defined as in Section 2.4 can be solved by reduction to a G^{2}TSP defined by a $G'=(V',E')$, Γ, Ξ and a V_g for every g in Γ, as follows:

As an intermediate step, one constructs a digraph $G''=(V'',E'')$. $V''=V\cup V'$ where V' is the set of all warehouses v_e for e in E_g of g in Γ. For every edge $e=(v,v',l,\bar{c})$ with v_e in V', one includes into E'' edges $(v,v_e,l,\bar{c})$ and $(v_e,v',\varepsilon,\bar{0})$, thereby explicitly representing that somewhere along e, there is warehouse v_e splitting e into two segments. For every two different vertices v and v' in V, one also includes into E'' one of the minimum-cost edges leading from v to v', if any.

G'' has the property that no pair of vertices is directly connected by more than one edge. For such a graph, the complexity of algorithms for finding a minimum-cost path $\bar{e}_{v,v'}$ between every two different vertices v and v' is $O(|V''|^3)$ or lower. For every two different v and v' in V', E' contains an edge $(v,v',\bar{e}_{v,v'},cost(\bar{e}_{v,v'}))$.

For every goal g in Γ, V_g is the set of all v_e with e in E_g. Solving the G^{2}TSP, one obtains a circuit $e_1,\ldots,e_k$. To extract the corresponding solution to the G^{2}RPP, one takes $lab(e_1)\cdot\ldots\cdot lab(e_k)$ and replaces every subsequence $(v,v_e,l,\bar{c}),(v_e,v',\varepsilon,\bar{0})$ in it with $(v,v',l,\bar{c})$.

If a G^{2}RPP is a GRPP, the procedure reduces it to a GTSP. While the general G^{2}TSP is left for further study, GTSP is discussed in Section 6.4.

6.3. Reducing Multi-Criteria Optimization to Single-Criterion Optimization

We have assumed that edge costs are vectors and demonstrated that multi-criteria optimization can be very helpful in TT construction. When we wanted to minimize traversal of undesirable edges, we didn't have to invent a special method like the one which Hierons employed in [26] to minimize traversal of reset edges; we simply extended cost vectors with another component.

The relation adopted for cost vector comparison possesses every property which one would expect for such a relation: For every $\bar{c}$ and $\bar{c}'$, $\neg(\bar{c}=\bar{c}')\wedge\neg(\bar{c}<\bar{c}')$ implies $\bar{c}'<\bar{c}$, $\bar{c}+\bar{0}=\bar{c}$ and $\bar{0}<\bar{c}'$ implies $\bar{c}<\bar{c}+\bar{c}'$. We therefore see no reason why a G^{2}TSP solver could not work directly with vector costs. Nevertheless, we are aware that typical G^{2}TSP solvers would not, because in the classical definition of GTSP, costs are non-negative reals. So in this section, we give a general method for encoding edge cost vectors into scalars.

The method has been inspired by [26]. It is based on the assumption that any cost vector $\bar{c}$ which a solver of a G^{2}TSP on a $G=(V,E)$ might attempt to handle is a $\sum_{e\in E'}cost(e)$ for an $E'\subseteq E$. We are looking for such a scalar encoding $scl(\bar{c})$ that for every $\bar{c}$ and $\bar{c}'$ in the adopted cost vector universe $C(G)$, $scl(\bar{c}+\bar{c}')=scl(\bar{c})+scl(\bar{c}')$ and $\bar{c}<\bar{c}'$ implies $scl(\bar{c})<scl(\bar{c}')$, so that the encoding cannot effect the choice of a circuit in G.

Let k denote the size of vectors in $C(G)$. For every $1\le i\le k$, let d_i denote the minimum element in $\{|cmp(\bar{c},i)-cmp(\bar{c}',i)|\ |(\{\bar{c},\bar{c}'\}\subseteq C(G))\wedge(cmp(\bar{c},i)\neq cmp(\bar{c}',i))\}\cup\{1\}$.

For every $\bar{c}$ in $C(G)$ and $1\le i\le k$, let $scl(\bar{c},i)=\sum_{i\le j\le k}w_j*cmp(\bar{c},j)$ denote the scalar encoding of $sfx(\bar{c},k-i+1)$, where $w_k=1/d_k$ and for every $1\le j<k$, $w_j=(1+\sum_{e\in E}scl(cost(e),j+1))/d_j$. $scl(\bar{c})$ is defined as $scl(\bar{c},1)$.

It is obvious that for every $1\le i\le k$, $scl((\bar{c}+\bar{c}'),i)=scl(\bar{c},i)+scl(\bar{c}',i)$. For a $\bar{c}=\sum_{e\in E'}cost(e)$ and a $\bar{c}'>\bar{c}$, let i denote the position in which the vectors start to differ. We now prove $scl(\bar{c}')>scl(\bar{c})$.

$scl(\bar{c}')-scl(\bar{c})=scl(\bar{c}',i)-scl(\bar{c},i)$
$=\sum_{i\le j\le k}w_j*cmp(\bar{c}',j)-\sum_{i\le j\le k}w_j*cmp(\bar{c},j)$
$=w_i*(cmp(\bar{c}',i)-cmp(\bar{c},i))+scl(\bar{c}',i+1)-scl(\bar{c},i+1)$
$\ge w_i*(cmp(\bar{c}',i)-cmp(\bar{c},i))-scl(\bar{c},i+1)$
$=(1+\sum_{e\in E}scl(cost(e),i+1))$
$\quad*((cmp(\bar{c}',i)-cmp(\bar{c},i))/d_i)-scl(\bar{c},i+1)$
$\ge 1+\sum_{e\in E}scl(cost(e),i+1)-scl(\bar{c},i+1)$
$=1+\sum_{e\in E}scl(cost(e),i+1)-scl(\sum_{e\in E'}cost(e),i+1)$
$=1+\sum_{e\in E}scl(cost(e),i+1)-\sum_{e\in E'}scl(cost(e),i+1)$
$=1+\sum_{e\in E\setminus E'}scl(cost(e),i+1)>0$

6.4. Solving GTSP

TSP is a well investigated problem for which numerous exact and approximate algorithms have been proposed. It therefore seems convenient to solve GTSP by reduction to TSP. Behzad and Modarres [11] suggest that a GTSP is first transformed into a modified version in which vertex clusters V_g are non-intersecting, there are no intra-cluster edges and the task is to find a circuit visiting exactly one vertex per cluster. They observe that such a transformation is easy [27], and show how the modified GTSP can be transformed into a TSP without increasing the number of vertices. Attempts

to solve GTSP by less conventional methods are also known, e.g. [28]. It is reasonable to expect that the research of GTSP will intensify, because there are a wide variety of real-world problems which can be modelled as a GTSP [29].

7. COPING WITH COMPLEXITY

GRPP or G^{2}RPP is, like RPP [12], NP-hard and the graph G on which we are solving it is typically enormous, because its subgraph G^* is enormous. Hence, to cope with the complexity, it might be convenient to work with a smaller G^* which is only an approximation of $unf(M^*)$. If Restrictions 1 to 5 are respected, systematic construction of such an approximation can proceed as follows:

One first looks in M^* for rooted tours satisfying individual predicates p in $PT(M^*)$, until one obtains such an initial rooted tour set IT that for a satisfactory subset PT^+ of $PT(M^*)$, IT for every predicate p in PT^+ contains a tour $\bar{\tau}$ with $p(\bar{\tau})$. While constructing an optimal tour satisfying a set of predicates is difficult, constructing any rooted tour satisfying an individual predicate is typically much easier.

For each rooted tour $\bar{\tau}$ in IT, one constructs $unf(\bar{\tau})$ and includes its vertices and edges into G^*. Completing the procedure for all $\bar{\tau}$ in IT, one obtains the first approximation of $unf(M^*)$. As by concatenating rooted tours satisfying individual predicates p in PT^+, one obtains a rooted tour satisfying all p in PT^+, one may as well decide that further enhancements of G^* are not necessary. It is, however, advisable that G^* is enhanced until its size reaches the acceptable upper limit or all rooted tours in M^* are covered, to encode additional information on possibilities for TT optimization, not only for optimizing its cost, but also for optimizing the subset of $PT(M^*)$ which it satisfies.

Constructing G^* as an approximation of $unf(M^*)$, one follows the precise unfolding procedure, except that instead of traversing M^* in all directions, one traverses it only along selected rooted tours, the ones which some heuristics identify as promising. Besides, it is acceptable that for a rooted tour $\bar{\tau}$ outside IT, the corresponding walk included into G^* is not $unf(\bar{\tau})$, but an approximation underestimating $PI(\bar{\tau}')$ of some prefixes $\bar{\tau}'$ of $\bar{\tau}$ and/or $PF(\bar{\tau}'')$ of some suffixes $\bar{\tau}''$ of $\bar{\tau}$. With Restrictions 2, 4 and 5, we know that such underestimation can do no harm. It only makes $\bar{\tau}$ a less attractive candidate for $\bar{\tau}^*$. On the other hand, it might help to keep the size of G^* within reasonable limits, by increasing the probability that two vertices from $unf(M^*)$ are mapped into the same vertex in G^*.

If Restrictions 1 to 5 are not respected, one is not allowed to construct $\bar{e}^*$ as a concatenation of rooted tours in G, implying that the strategy of focusing on rooted tours satisfying individual predicates p in PT^+ does not work. Besides, underestimation of a $PI(\bar{\tau}')$ or a $PF(\bar{\tau}'')$ is not always safe. Hence, one would have to proceed iteratively, repeatedly including into G^* additional promising tours from M^* and checking whether a satisfactory tour has already been covered.

The size of the constructed GRPP or G^{2}RPP can be further reduced by underestimating the stock of warehouses in G^*, thereby reducing E_g for individual products g in Γ, or even Γ itself. In an extreme case, a G^{2}RPP thereby becomes a GRPP, or a GRPP becomes an RPP (see Section 6.1).

8. ASSESSMENT OF CONTRIBUTIONS

As the paper pursues many research directions, it seems appropriate to summarize its contributions in a single place. In this section, we also provide additional critical assessment, to facilitate better judgement on when to use for DTGP the new method (or its individual specializations) and when to better stick to the well-established ones.

8.1. Contributions of Immediate Practical Interest

The proposed test generation method is generic, i.e., not sufficiently specific for immediate application in practice. However, when specialized with a specific well-defined testing strategy, it is a ready-to-use method. In this sense, we can say that we have developed *new test generation methods implementing, respectively, some of the best-known strategies for CTP* (see Sections 5.1, 5.2 and 5.3) *and a simple strategy for CTP supporting double-checking of states* (see Section 5.4). Unlike the earlier methods for the strategies, our methods implement the strategies precisely, in the sense that they *generate tests provably satisfying the adopted testing strategy to the maximum possible extent and at a minimum cost* (see Section 4.7).

One should consider the new methods whenever *seeking tests cheaper than those generated by the earlier methods*, which excel in their lower spatial and temporal complexity. A compromise would be to use the new methods with approximations (some are proposed in Section 7, and one can also resort to approximate solving of the GTSP to which the given DTGP reduces), carefully keeping the test generation process as close to its optimum course as the current circumstances allow.

As the proposed generic method accepts and precisely implements also the testing strategy which for the general DTGP leads to tests with the maximum possible discriminating power (see Section 5.8), we can say that we have developed *a method for systematic generation of absolutely optimal checking sequences*. Again, the above remarks on complexity apply, with the exception that this method is, unlike our methods mentioned above, not a specialization of the basic variant of the generic method described in Section 4, but of its variant described in Section 4.8, and is, hence, much more difficult to conduct in an approximate way (see Section 7).

Another feature in which the proposed methods excel is their ability to handle FSMs in which transition costs are vectors of costs ordered by their relative importance, i.e., the methods support *multi-criteria optimization*.

8.2. Contributions of Methodological Interest

We have *demonstrated that the nature of DTGP is that of GRPP* (see Section 5.8). This *explains why* the existing TT-generation methods typically *fail to produce a test of minimum cost*:

1) When choosing a testing strategy, those among them which are CT-generation methods systematically interpret DTGP as G^2RPP (see Section 5.7) and, consequently, adopt a strategy with a focus a priori narrowed on TTs of just some very specific kind. It seems that their reason for proceeding in such an approximate way is not only the wish to restrain the complexity of the TT-generation procedure, but to some extent also the common misconception that generation of optimal tests requires a fancy strategy, while the truth is just the opposite (see Section 5.8).

2) Regardless of how the methods interpret DTGP during strategy selection, during TT generation they solve DTGP as an RPP, but when a DTGP which is not an RPP is encoded into an RPP, one unavoidably loses some information on possibilities for TT optimization.

We have proposed a *new TT-generation method* (see Section 4) with the following convenient features:

1) The method is *generic, accepting as a parameter a wide class of testing strategies*, among them *the strategy leading to absolutely optimal tests* (see Section 5.8).

2) For the strategies it accepts, the method provides a *unified conceptual framework* and a *uniform encoding*, thereby *facilitating* their *comparison*. The earlier TT-generation methods inconveniently differ not only in the adopted testing strategies, but also in their encoding. The *encoding* we employ is *unusually simple and intuitive for a method supporting subtest overlapping*. One should, hence, consider using the method whenever trying to implement a new strategy involving subtest overlapping.

3) With the method, *strategy specification* proceeds in a purely *declarative* manner. One simply specifies, in a very *abstract* way, the *qualitative* properties which TT is supposed to possess.

4) For any strategy it accepts, the method *automatically generates a test provably satisfying the adopted requirements to the maximum possible extent and at a minimum cost* (see Section 4.7). This is achieved by *handling all optimization concerns in a fully integrated manner*.

5) The method solves a given DTGP by reducing it to an instance of some generic graph-theoretic problem. With this approach, it *strictly separates domain-specific and general graph-theoretic concerns*.

6) It supports *transition costs which are vectors* of costs ordered by their relative importance. With this feature, one can, for example, *directly encode additional requirements* of testing strategies *for avoidance of transitions* whose traversal could decrease the discriminating power of the test (see Section 5.6). The method can nevertheless benefit from tour-construction tools accepting only graphs with scalar edge costs, for we have provided a *procedure for encoding cost vectors into equivalent scalar costs* (see Section 6.3).

7) The method can be employed not only for *transition-oriented*, but also for *fault-model-driven* test construction (see Section 5.8).

With the method, we have tried to set *an example of how to develop new TT-generation algorithms in an orderly manner*. As TT-generation algorithms have to search really vast spaces, they tend to be heuristic, i.e., approximative. To find good approximations, it is important to know what is being approximated. An ideal scheme for developing TT-generation algorithms would, hence, be the following:

1) Conceive an *exact model of the given problem*, encoding *all the information necessary for producing an optimal solution*. For example, the graph G from Section 4, if constructed under the testing strategy of Section 5.4 and adequately interpreted as a graph hosting warehouses, is an exact model for CTP with double-checking of states (CTPDC).

2) Finding an optimal solution for a problem is a search problem on its model. Identify the *exact nature of the search problem* and conceive a procedure for its *exact solving*. In the example above, the original problem is CTPDC, while the associated search problem is the well-investigated RPP (see Section 6.1).

3) To cope with the cases where the *complexity of the model* is problematic, identify various interesting classes of *approximate models*, characterizing each class by its potential to lead to a high-quality solution and by the complexity of the associated search problem. For CTPDC, the strategy of Section 5.1 produces an approximate model which typically leads to a less powerful TT, while the associated search problem is, interestingly, in a general case a GRPP (see Section 3.2), i.e., of a more complex kind than the one associated with the precise model of CTPDC.

4) To cope with the cases where the *complexity of the search problem* associated with the exact model is problematic, try to identify various interesting less complex *specializations of the search problem*, focusing on those to which various interesting sub-optimal strategies translate the original problem. For example, Aho et al. [5], knowing that RPP is NP-hard, focused on its specialization Rural Chinese Postman Problem (RCPP), simultaneously developing a polynomial-time procedure for RCPP and demonstrating that under the testing strategy in which final states of transitions are checked with UIO sequences, in the very common cases where every state in M has a transition to the root or every state has a loop, CTP reduces to RCPP.

5) One might as well resort to various *approximations of the search procedure*. For example, after precisely reducing a GRPP to a GTSP, one might resort to approximate GTSP solving. However, as such approximations do not pay regard to the specifics of the original problem, they often unexpectedly neglect some information which they should not. Therefore, if some information can be safely neglected, its neglecting should better be implemented as its omission from the problem model, and not as imprecise search of the model.

9. SUGGESTIONS FOR FURTHER WORK

To get the full benefit of the proposed integrated approach, it would be necessary to find CPSs for various existing testing strategies. As this might reveal the need for *a broader class of predicate systems*, we are currently generalizing the method to predicate systems whose manipulation requires multiple traversals of M^*.

It seems that the method naturally generalizes to *non-deterministic* M, since our concept of a testing strategy being just a statement on the required properties of the observed I/O sequence does not depend on M being deterministic. It is just that with a non-deterministic M, one must be aware that there might be multiple legal responses to a test, where each response must provide the required evidence of correctness. Hence, after a testing strategy has been encoded into a graph, the task is not to find an optimal individual tour on the graph, as for DTGP, but an optimal *tree* of tours, possibly encoding an *adaptive* test.

For all strategies, old and new, it is important to identify *interesting approximations*, from now on, hopefully, in a more systematic manner and with a more uniform encoding.

For all strategy implementations, exact and approximate, it is important to identify the associated *generic search problems* and stimulate production of tools for their *efficient solving*, so that people working in the field of testing can focus entirely on the domain-specific aspects of their problems.

REFERENCES

[1] Hierons, R. M. (2006) Separating sequence overlap for automated test sequence generation. *Automated Software Engineering*, **13**(2), 283–302.

[2] Chen, M.-S., Choi, Y., and Kershenbaum, A. (1990) Approaches utilizing segment overlap to minimize test sequences, *Proc. PSTV'90*, Ottawa, Canada, 12–15 June, pp. 85–98, North-Holland, Amsterdam.

[3] Shen, N. Y., Lombardi, F., and Dahbura, A. T. (1990) Protocol conformance testing using multiple UIO sequences. *IEEE Transactions on Communications*, **40**(8), 1282–1287.

[4] Yang, B., and Ural, H. (1990) Protocol conformance test generation using multiple UIO sequences with overlapping. *Proc. ACM SIGCOMM'90*, Philadelphia, PA, 24–27 September, pp. 118–125, ACM.

[5] Aho, A. V., Dahbura, A.T., Lee, D., and Uyar, M. U. (1991) An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours. *IEEE Transactions on Communications*, **39**(11), 1604–1615.

[6] Miller, R. E., and Paul, S. (1993) On the generation of minimal length conformance tests for communications protocols. *IEEE/ACM Transactions on Networking*, **1**(1), 116–129.

[7] Ural, H., and Zhu, K. (1993) Optimal length test sequence generation using distinguishing sequences. *IEEE/ACM Transactions on Networking*, **1**(3), 358–371.

[8] Choi, Y., Kim, D., Kim, J., Park, Y., and Chung, I. (1995) Protocol test sequence generation using UIO and BUIO. *Proc. IEEE ICC'95*, Seatle, WA, 18–22 June, Vol. 1, pp. 362–366, IEEE Communications Society.

[9] Hierons, R. M. (1996) Extending test sequence overlap by invertibility. *The Computer Journal*, **39**(4), 325–330.

[10] Hierons, R. M. (1997) Testing from a finite state machine: Extending invertibility to sequences. *The Computer Journal*, **40**(4), 220–230.

[11] Behzad, A., and Modarres, M. (2002) A new efficient transformation of the generalized traveling salesman problem into traveling salesman problem. *Proc. ICSE'02*, Las Vegas, NV, 6–8 August.

[12] Lenstra, J. K., and Kan, A. H. G. R. (1976) On general routing problems. *Networks*, **6**, 273–280.

[13] Chow, T. S. (1978) Testing software design modelled by finite state machines. *IEEE Transactions on Software Engineering*, **4**(3), 178–187.

[14] Fujiwara, S., von Bochmann, G., Khendek, F., Amalou, M., and Ghedamsi, A. (1991) Test selection based on finite state models. *IEEE Transactions on Software Engineering*, **17**(6), 591–603.

[15] Rezaki, A., and Ural, H. (1995) Construction of checking sequences based on characterization sets. *Computer Communications*, **18**(12), 911–920.

[16] Shen, X. J., Scoggins, S., and Tang, A. (1991) An improved RCP-method for protocol test generation using backward UIO sequences. *Proc. ACM/IEEE.CS SAC'91*, Kansas City, MO, 3–5 April, pp. 284–293, IEEE Computer Society Press.

[17] Hennie, F. C. (1964) Fault-detecting experiments for sequential circuits. *Proc. Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, Princeton, NJ, 11–13 November, pp. 95–110, IEEE.

[18] ISO/IEC 13211 (1995) Information technology – Programming languages – Prolog. International Organization for Standardization, Geneva, Switzerland.

[19] Guo, Q, Hierons, R. M., Harman, M., and Derderian, K. (2006) Improving test quality using robust unique input/output circuit sequences (UIOCs). *Information and Software Technology*, **48**(8), 696–707.

[20] Anido, R., and Cavalli, A. (1995) Guaranteeing full fault coverage for UIO-based testing methods. *Proc. IWPTS'95*, Evry, France, 4–6 September, pp. 221–236, Chapman & Hall, London.

[21] Inan, K., and Ural, H. (1999) Efficient checking sequences for testing finite state machines. *Information and Software Technology*, **14**(11–12), 799–812.

[22] Hierons, R. M, and Ural, H. (2002) Reduced length checking sequences. *IEEE Transactions on Computers*, **51**(9), 1111–1117.

[23] Chen, J., Hierons, M. R., Ural, H., and Yenigün, H. (2005) Eliminating redundant tests in a checking sequence. *Proc. TestCom 2005*, Montreal, Canada, 31 May – 2 June, pp. 23–39, LNCS 3502, Springer-Verlag, Berlin.

[24] Tekle, K. T., Ural, H., Yalcin, M. C., and Yenigün, H. (2005) Generalizing redundancy elimination in checking sequences. *Proc. ISCIS'05*, Istanbul, Turkey, 26–28 October, pp. 915–926, LNCS 3733, Springer-Verlag, Berlin.

[25] Hierons, R. M., and Ural, H. (2006) Optimizing the length of checking sequences. *IEEE Transactions on Computers*, **55**(5), 618–629.

[26] Hierons, R. M. (2004) Using a minimal number of resets when testing from a finite state machine. *Information Processing Letters*, **90**(6), 287–292.

[27] Lien, Y., Ma, E., and Wah, B. W. (1993) Transformation of the generalized traveling salesman problem into the standard traveling salesman problem. *Information Sciences*, **74**(1–2), 177–189.

[28] Huang, H., Yang, X. W., Hau, Z. F., Wu, C. G., Liang, Y. C., and Zha, X. (2005) Hybrid chromosome genetic algorithm for generalized traveling salesman problems. *Proc. ICNC'05*, Changsha, China, 27–29 August, pp. 137–140, LNCS 3612, Springer-Verlag, Berlin.

[29] Laporte, G., Asef-Vaziri, A., and Srikandarajah, C. (1996) Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society*, **47**(12), 1461–1467.