Optimizing Probabilities of Real-Time Test Case Execution

Nicolás Wolovick and Pedro R. D'Argenio
Fa.M.A.F.
Universidad Nacional de Córdoba
5000 Córdoba, Argentina
Email: {nicolasw,dargenio}@famaf.unc.edu.ar

Hongyang Qu
Department of Computing
Imperial College
London SW7 2RH, U.K.
Email: hongyang.qu@imperial.ac.uk

***Abstract*—Model-based test derivation for real-time system has been proven to be a hard problem for exhaustive test suites. Therefore, techniques for real-time testing do not aim to exhaustiveness but instead respond to particular coverage criteria. Since it is not feasible to generate complete test suites for real time systems, it is very important that test cases are executed in a way that they can achieve the best possible result. As a consequence, it is imperative to increase the probability of success of a test case execution (by 'success' we actually mean 'the test finds an error'). This work presents a technique to guide the execution of a test case towards a particular objective with the highest possible probability. The technique takes as a starting point a model described in terms of an input/output stochastic automata, where input actions are fully controlled by the tester and the occurrence time of output action responds to uniform distributions. Derived test cases are sequences of inputs and outputs actions. This work discusses several techniques to obtain the optimum times in which the tester must feed the inputs of the test case in order to achieve maximum probability of success in a test case execution. In particular, we show this optimization problem is equivalent to maximizing the sectional volume of a convex polytope when the probability distributions involved are uniform.**

I. Introduction

Testing is an essential component of the life cycle of computer systems. It is usually at the last stage of the development process and it has the aim to find the errors that were introduced in the implementation or that remained undetected in previous verification phases. One prominent approach to test derivation is the so-called *model-based testing*, which is mainly a black-box technique (see, e.g., [6]). In this approach, tests are derived from a model that provides an abstract description of the functional and quantitative aspects of the system under test. A widely studied approach is test derivation out of automata-like models (e.g. [18], [28]).

In the context of this work we focus on testing real-time systems. Model-based test derivation for real-time system has been proven to be a hard problem for exhaustive test suites [27]. Therefore, techniques for real-time testing do not aim to exhaustiveness and respond to particular coverage criteria (see e.g. [7], [22], [10], [5] and references therein).

Since it is not feasible to generate complete test suites for real time systems, it is important that test cases are executed in a way that they can achieve the best possible result. As tests cannot guarantee the absence of error, one may consider that a successful test is the one that actually finds an error. This work presents a technique to guide the execution of a test case towards a particular objective with the highest possible probability. Such objective very much depends on the coverage criteria and the aim of the test case. For example, one may argue that errors are more likely to happen around the boundaries of time specifications [7]. So one would like to guide a test to increase the probability of reaching a particular boundary. Alternatively, one may like to increase probabilities for stress testing, thus guiding the test to reach extreme situations that are more likely to fail (e.g. buffer overflows).

In this work, we assume that the system is modelled as a set of parallel components. Each component is modelled as a controlled automaton where (controlled) input actions are governed by the environment and (uncontrolled) output actions can occur within a given time interval. The best possible choice to estimate the occurrence time of an event when only the time interval is known is by assuming that it is uniformly distributed (uniform distributions are precisely the maximum entropy probability distribution supported in a closed interval). There is also an important technical consequence of choosing only uniform distributions: the density function of a uniform is a constant function, and hence the problem we deal with is much more tractable. Therefore, the technique starts with a model of the real-time system, represented in a controlled variation of the so-called generalized semi-Markov processes (GSMP) [12]. In this model there are (controlled) input and (uncontrolled) output actions. Inputs are fully controlled by the user and outputs are temporized actions controlled by a uniform distribution just like in GSMPs.

It is *not the aim* of this paper to address test case derivation in general, but *only* to derive the best possible times in which inputs should be fed. So we assume test cases are already obtained somehow on (some abstraction of) this model (see Sec. IV). We consider the test case to be a sequence of inputs and outputs ending on a verdict. The execution of the test case

Research supported by the CNRS/CONICET Cooperation Project "Verification Methods for Timed and Probabilistic Concurrent Programs". N. Wolovick and P.R. D'Argenio are supported by the the ANPCyT project PICT 26135 and CONICET project PIP 6391. P.R. D'Argenio is also a CONICET Researcher. H. Qu is supported by the European Commission Framework 6 funded project CONTRACT (IST Project Number 034418).

can be seen as a game between the tester, that feeds the inputs, and the system under test (SUT), that plays an adversarial role by producing outputs. These outputs may be expected in the test cases, or unexpected, in which case the verdict would most probably be inconclusive.

We present a technique to derive the optimal time to feed each input of the test case to the SUT so that the system is guided with highest probability towards the end of the test case, where the actual verdict takes place. Most of the paper is devoted to the calculations of the input times *off-line*, that is, prior to the actual execution of the test. We show that the problem of computing the optimal input transition times to maximize the probability of a test case execution, is equivalent to maximizing the sectional volume of a convex polytope. This actually presents the bulk of the technical details. The basic ideas of the technique are based on [14], where test cases were merely observational (not controlled), and hence present only a calculational technique, in the sense that it only reports a probability value (contrarily to our work that attacks an optimization problem).

Of course, fixing the time in which every input should be feed prior to the execution of the test is suboptimal. Consider, for instance, a test *aibjc* where i and j are inputs and the rest outputs. It is most likely that the optimal time to feed j depends on all the past history, that is, in order to decide when to feed j it is better to know the occurrence times of a, i, and b. Therefore it is convenient to be adaptive, that is, to select the input time only when the execution time of all previous events are known. We also present an adaptation of the previous algorithm to an *on-the-fly* version.

As computation takes time and we are testing real-time systems, on-the-fly calculations are continuously racing against the occurrence time of the SUT output events. Therefore, we discuss a modification of this "ideal" on-the-fly algorithm to deal with the races and get the best possible choice on the time to feed an input.

These novel ideas leave room for much improvement. Some of the used algorithms are of exponential nature but used together with simple DBM-like structures [8]. Moreover, some calculations in the on-the-fly algorithm can be partly reused along the execution of the test and we have not explored how much of the calculations can be performed symbolically off-line. These and many other issues are discussed in the conclusion of the paper.

Related topics: In [20] optimal input times are also selected but in a different context and model. They obtain a parametric integral that is a measure of the correctness of the implementation. Besides the stochastic component is separated from the model. The non-observable timeouts of a timed finite state machine are modelled through an inertia probability function. Although similar integrals are obtained, in this work it is not mentioned how to obtain the optimal times.

Organization of the paper: Sec. II introduces the preliminaries on mathematics. The model is introduced in Sec. III and Sec. IV discusses possible solutions to test case derivation. Sec. V presents the calculation of the probability of a test case. This probability value is actually a function on the occurrence time of the inputs and it is the object of maximization. In fact, such function is a *parametric volume* which is subject of study in Sec. VI. The maximization algorithm is presented in Sec. VII. Sec. VIII discusses the on-the-fly maximization algorithm and the variant that deals with the race against the occurrence time of the SUT output events. Sec. IX concludes the paper mainly focusing on possible improving techniques and further work.

II. Mathematical background

Given a set Ω and a collection $\mathcal{F}$ of subsets of Ω, we call $\mathcal{F}$ a *σ-algebra* iff $\Omega \in \mathcal{F}$ and $\mathcal{F}$ is closed under complement and denumerable disjoint union. We call the pair $(\Omega, \mathcal{F})$ a *measurable space*. Any set $A \in \mathcal{F}$ is called *measurable*. By $\mathcal{B}(T)$ we denote the Borel σ-algebra of topological space T generated by open sets, e. g., $\mathcal{B}(\mathbb{R}^+)$ denotes the set generated by all right-open intervals over $\mathbb{R}^+$. Given a measurable space $(\Omega, \mathcal{F})$, a σ-additive function $P : \mathcal{F} \to [0, 1]$ is called *probability measure* if $P(\Omega) = 1$. The triple $(\Omega, \mathcal{F}, P)$ is a *probability measure space*. A function $f : (\Omega_1, \mathcal{F}_1) \to (\Omega_2, \mathcal{F}_2)$ is *measurable* if $\forall A_2 \in \mathcal{F}_2,\ f^{-1}(A_2) \in \mathcal{F}_1$, i. e., the inverse function maps measurables to measurables. This condition is fulfilled for a Borel σ-algebra over the reals iff $f^{-1}([a, b)) \in \mathcal{B}(\mathbb{R})$, i. e. it has to be valid for the generators. A *random variable* X on a probability space $(\Omega, \mathcal{F}, P)$ is a Borel measurable function from Ω to $\mathbb{R}$, for example, the identity function or any linear function is a random variable on the probability space $(\mathbb{R}, \mathcal{B}(\mathbb{R}), P)$. Given random variable X, the *probability measure induced* by X is $P_X(B) \doteq P\{\omega \mid X(\omega) \in B\} = P(X^{-1}(B))$, where $B \in \mathcal{B}(\mathbb{R})$. A Borel measurable function f is the *probability density* of random variable X if $P_X(B) = \int_B f(x)\ \mathrm{d}x$ for all $B \in \mathcal{B}(\mathbb{R})$. Examples of a density function is the *uniform* distribution on $[a, b]$, $f(x) = (\textbf{if } a \le x \le b \textbf{ then } (b - a)^{-1} \textbf{ else } 0)$. A *random vector* X is a measurable function from Ω to $\mathbb{R}^n$, which all the above definitions apply and it can be regarded as n random variables $(X_1, \ldots, X_n)$. These random variables are said to be *independent* if $P\{X_1 \in B_1, \ldots, X_n \in B_n\} = P\{X_1 \in B_1\} \cdots P\{X_n \in B_n\}$ and this corresponds to the intuition that knowledge about any subset of random variables $\{X_i\}$ does not affect the odds of the rest. Given an independent random vector $X = X_1, \ldots, X_n$ with density functions $f_1, \ldots, f_n$, by previous definition and [1, Corollary 4.8.5] we have:

$$P_X(B) = \int_B f_1(x_1) \ldots f_n(x_n)\ \mathrm{d}x_1 \ldots\ \mathrm{d}x_n \tag{1}$$

A subset K of Euclidean space $\mathbb{R}^n$ is said to be *convex* if for all $x_1, x_2 \in K$ and $0 < \lambda < 1$ then $\lambda x_1 + (1 - \lambda)x_2 \in K$. In $\mathbb{R}^n$ we call *half-space* the sets of points in one of the two regions in which a *hyperplane* divides the whole space, i. e. the solutions of $a_1 x_1 + \cdots + a_n x_n \le b$. A *convex polytope* is a bounded subset of $\mathbb{R}^n$ which can be equivalently defined as the solutions of the inequality $A\vec{x} \le \vec{b}$ or the finite intersection of half-spaces (This is the so-called *halfspace representation*.). Convex polytopes are *closed under intersections* and *projections*. Given two

polytopes K_1, K_2 the *Minkowski sum* is the sum of all vectors belonging to each polytope $K_1 + K_2 \doteq \{x_1 + x_2 \mid x_1 \in K_1, x_2 \in K_2\}$. We could seemingly define *scalar multiplication* λK of a polytope. By $Vol(K)$ we denote the *volume* of polytope K.

III. The model

Definition 1: A *Stochastic Timed I/O Automaton (STIOA)* is a structure $\mathcal{T} = (V, \Sigma)$ where

1) V is a finite set of *program variables*, and
2) Σ is a finite set of *transitions* partitioned in two sets: set Σ_I of *input transitions* and set Σ_O of *output transitions*. Each transition $\alpha \in \Sigma$ includes the following components:
   a) A first order predicate $en(\alpha)$ over the variables V called *enabling condition of* α.
   b) A transformation function $F_\alpha : V \to V$ on the program variables V which is applied to them if transition α is taken.
   c) For output transitions, the interval $[l_\alpha, u_\alpha]$ states the period during which α can be taken starting when $en(\alpha)$ first holds. We require $l_\alpha < u_\alpha$, and for simplicity we set bounds in $\mathbb{N}$.

Each output transition α is associated with a *count-down clock* $cl(\alpha)$. Every time α becomes enabled, $cl(\alpha)$ is set to a value in the interval $[l_\alpha, u_\alpha]$, and its value starts to decrease. Once it is equal to zero, α is executed. If α is disabled before its clock reading reaches zero, the clock stops running and it is reset to zero (note that α is not executed). There is also an increasing *global clock* $cl(gt)$ per system. When the system starts to run, it begins to run and never stops. For $\alpha \in \Sigma_O$, $cl(\alpha)(s)$ (respectively $cl(gt)(s)$) denotes the reading of $cl(\alpha)$ (respectively $cl(gt)$) in the state s (value of variables and clocks), and for $\alpha \in \Sigma_O \cup \Sigma_I$, $s \models en(\alpha)$ represents that the enabling condition $en(\alpha)$ of α holds in the state s, i. e., α is enabled in s.

The model is similar to the one in [14]. However, we differentiate between input and output transitions, obtaining a nondeterministic system. Output transitions are not governed by the environment and the occurrence time of an output transition $\alpha_O \in \Sigma_O$ respects a probability distribution whose support set falls in the interval $[l_{\alpha_O}, u_{\alpha_O}]$. For the rest of the work, we fix this distribution to be uniform in such interval.

The input transitions are controlled by the environment. The precise moment in which an input transition takes place is chosen non-deterministically by the environment. Hence it does not obey any probability distribution inherent to the system. Basically these inputs parametrize the model in [14].

We define probabilistic execution as an alternating sequence of time passing and transition executions starting from an initial state s_0.

Definition 2: A *probabilistic execution* of a system $\mathcal{T}$ is a finite sequence of the form $s_0 g_1 \alpha_1 s_1 g_2 \alpha_2 s_2 g_3 \ldots s_{n-1} g_n \alpha_n s_n$, where s_i, g_i are states and α_i are transitions. The state s_0 is the initial state of $\mathcal{T}$. The adjacent pair s_i, g_{i+1} represents *time passing*, therefore $V(s_i) = V(g_{i+1})$, whereas all the enabled clocks move at the same speed, e.g., $cl(\alpha_{i+1})(s_i) - cl(\alpha_{i+1})(g_{i+1}) = cl(gt)(g_{i+1}) - cl(gt)(s_i)$ if $\alpha_{i+1} \in \Sigma_O$. The triple $g_i \alpha_i s_i$ represents a *transition execution*, so α_i has to be enabled in g_i, i. e., $g_i \models en(\alpha_i)$ and, if $\alpha_i \in \Sigma_O$, its clock reading should be 0, that is, $cl(\alpha_i)(g_i) = 0$. The transition execution is instantaneous such that $cl(gt)(g_i) = cl(gt)(s_i)$, and the state changes accordingly $V(s_i) = F_{\alpha_i}(V(g_i))$. If a transition $\beta \in \Sigma_O$ becomes enabled by the execution of α_i such that $g_i \models \neg en(\beta) \wedge s_i \models en(\beta)$, or $\beta = \alpha_i$ becomes enabled again $s_i \models en(\beta)$, then $cl(\beta)(s_i)$ is a value chosen randomly in $[l_\beta, u_\beta]$. In the case a transition $\gamma \in \Sigma_O$ is disabled by the execution of α_i such that $g_i \models en(\gamma) \wedge s_i \models \neg en(\gamma)$, its clock is reset, i. e., $cl(\gamma)(s_i) = 0$. Clocks of other transitions remain unchanged in s_i and g_i.

Definition 3: A *path* is a finite sequence of actions $\sigma = \alpha_1 \ldots \alpha_n$. It is *consistent* with an execution ρ if it is obtained by removing all the state components of ρ.

Participating transitions: When the system reaches a state, it defines a set of transitions that are enabled. At this point, a race begins if there are more than one enabled transition. Therefore the probability of taking a transition is a function of all enabled transitions, not only the one that is needed to be taken.

Definition 4: The *enabledness period* of a transition α_i (not necessarily in ρ) with respect to the execution $\rho = s_0 g_1 \alpha_1 s_1 g_2 \alpha_2 s_2 g_3 \ldots s_{n-1} g_n \alpha_n s_n$ is $[\mathsf{start}(i), \mathsf{stop}(i)]$, where $\mathsf{start}(i) \doteq \min\{j \mid 0 \leq j \leq (n-1) \wedge s_j \models en(\alpha_i)\}$ and $\mathsf{stop}(i) \doteq \max\{j \mid 1 \leq j \leq n \wedge g_j \models en(\alpha_i)\}$. If a transition is never enabled in ρ, its enabledness period is *empty*.
A transition is a *participating transition* of an execution ρ if its enabledness period with respect to ρ is nonempty. By definition α_i in ρ has nonempty enabledness period that includes i. We consider all transitions participating in a path to be different, renaming each occurrence when necessary to comply with this restriction. Then we can easily see that the enabledness period of α_i in ρ is continuous, i. e. $s_{\mathsf{start}(i)} \models en(\alpha_i)$, $g_{\mathsf{stop}(i)} \models en(\alpha_i)$ and $\forall j : j \in [\mathsf{start}(i) + 1, \mathsf{stop}(i) - 1] : s_j \models en(\alpha_i) \wedge g_j \models en(\alpha_i)$.

IV. Issues on test case derivation

It is not the aim of this paper to address test case derivation. Nevertheless, in this section we briefly review existing techniques and mention how they can be applied to our model.

First of all, notice that a STIOA is a (externally) deterministic I/O labelled transition system (IOTS) when time aspects are disregarded: states are given only by assignment on variables in V (but have no information on clock values), the set of labels is $\Sigma_I \cup \Sigma_O = \Sigma$, and the transition relation is defined by $s \xrightarrow{\alpha} s'$ whenever $s \models en(\alpha)$ and $V(s') = F_\alpha(V(s))$ for all $\alpha \in \Sigma$. The IOTS can be easily modified to be input enabled. A possible modification is to add a new state s_δ and a transition $s \xrightarrow{\alpha} s_\delta$ for each state s and $\alpha \in \Sigma_I$ such that $s \not\models en(\alpha)$.

On this model, it is possible to use, for instance, Tretmans' algorithm for test case derivation [28]. Test cases obtained this way are trees where each node has either a single outgoing input transition, or a set of outgoing outputs. As we need tests

that are linear sequences, they can be obtained from the tree by selecting a descending path. A transition that branches away from the path yields a failing test if it ends in a failing node in the original test, or an inconclusive test if the original test has a continuation.

A STIOA can also be translated to observable nondeterministic finite state machine (ONFSM) by taking two new labels $-_i$ and $-_o$ that identify the null input and null output respectively, and defining $s \xrightarrow{\alpha/-_o} s'$ if $s \models en(\alpha)$ and $V(s') = F_\alpha(V(s))$ for all $\alpha \in \Sigma_I$; if, instead, $\alpha \in \Sigma_O$, define $s \xrightarrow{-_i/\alpha} s'$. The set of states are as for IOTS. On this model, test cases can also be derived (see, e.g, [18]).

Other methods and techniques are possible, but, in any case, the generated test cases should be carefully considered. When the time behavior is disregarded, impossible test cases can be derived. For instance, consider the STIOA

$$\bullet s_1 \xleftarrow[a]{[1,2]} \bullet s_0 \xrightarrow[b]{[3,4]} \bullet s_2$$

with s_0 being the initial state. A test for the b transition is never realized since transition a is always faster. If time is taken into account in order to derive tests, impossible tests are not generated (see e.g. [7], [27], [22], [10], [4]), though timed test derivation may become prohibitively expensive if trying to be exhaustive [27].

A successful test is the one that actually finds an error. In the following, we present a technique to probabilistically guide the execution of a test case towards a particular objective. Such objective very much depends on the coverage criteria and the aim of the test case.

V. The probabilities of a test case

We first define a *test case* as a path ending in an output action, i. e. given $\sigma = \alpha_1\alpha_2 \dots \alpha_n$, then $\alpha_n \in \Sigma_O$. The assumption that test cases end with an output is reasonable (inputs can be fed at any moment) and consistent with existing techniques for test derivation (see e.g. [28]). Given a test case running on the execution $\rho = s_0 g_1 \alpha_1 s_1 g_2 \alpha_2 s_2 g_3 \dots s_{n-1} g_n \alpha_n s_n$, let $[\rho] = \{\alpha_1, \dots, \alpha_n\}$ be the set of transitions in the test case and $[\rho]' = \{\alpha_{n+1}, \dots, \alpha_m\}$ be the set of transitions that are enabled somewhere along ρ but do not belong to $[\rho]$, i. e., they are not executed by ρ. We do not consider input transitions that are enabled along ρ but deviate from its execution. Such inputs would make the test case render an inconclusive verdict something one can avoid since inputs are controllable.

Let $[\rho]_I = \{I_1, \dots, I_k\} \subseteq [\rho] \cup [\rho]'$ be the set of the input transitions and $[\rho]_O = \{O_1, \dots, O_l\} \subseteq [\rho] \cup [\rho]'$ the set of the output transitions, where $k + l = m$. Let $X = \{x_0, \dots, x_n\}$ be the set of global time points where $x_i = cl(gt)(s_i)$ $(1 \le i \le n)$ is the instant when $\alpha_i \in [\rho]$ is taken (we take $x_0 = 0$). For any output transition $\alpha_{O_j} \in [\rho]_O$ with the enabledness period $[\mathsf{start}(O_j), \mathsf{stop}(O_j)]$, $x_{\mathsf{start}(O_j)} \in X$ is the time point where α_{O_j} becomes enabled, and $x_{O_j} = x_{\mathsf{start}(O_j)} + cl(\alpha_{O_j})(s_{\mathsf{start}(O_j)})$ is the time α_{O_j} is expected to occur, where $cl(\alpha_{O_j})(s_{\mathsf{start}(O_j)})$ is the initial value of the clock $cl(\alpha_{O_j})$ in the state $s_{\mathsf{start}(O_j)}$. For any $\alpha'_{O_j} \in [\rho]'$, $x_{\mathsf{stop}(O_j)} \in X$ is the time α'_{O_j} becomes disabled.

In order to compute the probability of executing the given test case, we need to set up time constraints for the test case. First of all, the global time points in X are ordered in time, which is deduced from the sequential execution of transitions in $[\rho]$.

$$x_0 < x_1 < \dots < x_n \tag{2}$$

For every output transition (taken or enabled but not taken) $\alpha_{O_j} \in [\rho]_O$, the initial value of the clock $cl(\alpha_{O_j})$ has to be chosen within its bounds when it has been enabled.

$$l_{O_j} \le cl(\alpha_{O_j})(s_{\mathsf{start}(O_j)}) = x_{O_j} - x_{\mathsf{start}(O_j)} \le u_{O_j} \tag{3}$$

To make sure that all transitions $\alpha_{O_j} \in [\rho]'$ are not executed, we require the following constraint, which means that when α_{O_j} becomes disabled, it has a nonzero clock value.

$$x_{\mathsf{stop}(O_j)} < x_{O_j} \tag{4}$$

Note that if the system is composed solely of outputs, then all x_{O_j} would be bounded. Adding inputs can invalidate this property. Since we are aiming to give the best timing to feed inputs we can safely disregard systems where this time could be arbitrary. From now on, we will assume that the STIOA model and the test case define an inequality system where all global time values $x_1, \dots, x_m$ are bounded.

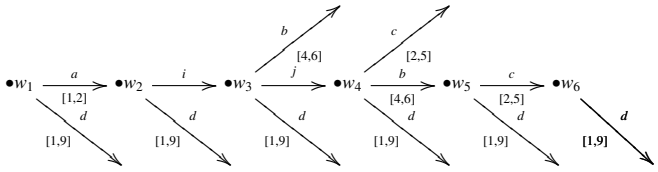
Example: This system has three parallel components with two inputs i, j and four output transitions a, b, c, d. The output transition a is used to synchronize the enabling of i and j, and for that purpose it was split in a synchronized sender $a!$ and a receiver $a?$.

$$\bullet s_1 \xrightarrow{a?} \bullet s_2 \xrightarrow{i} \bullet s_3 \xrightarrow[[4,6]]{b} \bullet s_4$$

$$\bullet t_1 \xrightarrow[[1,2]]{a!} \bullet t_2 \xrightarrow{j} \bullet t_3 \xrightarrow[[2,5]]{c} \bullet t_4$$

$$\bullet u_1 \xrightarrow[[1,9]]{d} \bullet u_2$$

The relevant part of the parallel composition of this system with respect to the test case $\sigma = aijbc$ is:



Note how transitions b, c, d have enabledness period that are not singletons therefore its clocks are set in its first occurrence. Below is the summary of all inequalities obtained using (2) and (4) for the left column, and (3) for the right column.

$$\begin{array}{ll} 0 < x_a < x_i < x_j < x_b < x_c & 1 \le x_a \le 2 \\ x_c < x_d & 4 \le x_b - x_i \le 6 \\ & 2 \le x_c - x_j \le 5 \\ & 1 \le x_d \le 9 \end{array}$$

□

Given the execution ρ, inequalities (2), (3) and (4) define a region B ranging on two different types of variables: inputs x_{I_i} and outputs x_{O_j}. The inputs are defined by the environment therefore we consider them as external parameters.

The outputs have a stochastic behavior given by independent random variables on the probability space $(\Omega, \mathcal{F}, P) = ([0,1], \mathcal{B}([0,1]), U[0,1])$, i. e. the standard uniform probability space on the real interval $[0,1]$. Each random variable is in fact a measurable linear function $x_{O_j} : [0,1] \to \mathbb{R}^+$ that maps the interval $[0,1]$ to lay in (3):

$$x_{O_j}(w) = (u_{O_j} - l_{O_j})w + l_{O_j} + x_{\mathsf{start}(O_j)} \tag{5}$$

Region B can be seen as a function from inputs parameters to a region ranged by independent uniformly distributed random variables, $B(x_{I_1}, \dots x_{I_k})$. The probability of this *parametric region* B is given by equation (1).

$$P_{x_{O_1},\dots,x_{O_l}}(B(x_{I_1},\dots,x_{I_k})) = \int_{B(x_{I_1},\dots,x_{I_k})} f_1(x_{O_1}) \dots f_l(x_{O_l})\, \mathrm{d}x_{O_1} \dots\, \mathrm{d}x_{O_l}$$

where f_j is the probability density distribution of random variable x_{O_j}, that is $((x_{\mathsf{start}(O_j)} + u_{O_j}) - (x_{\mathsf{start}(O_j)} + l_{O_j}))^{-1} = \frac{1}{u_{O_j} - l_{O_j}}$. Factoring out constants, the probability of an execution is:

$$P_{x_{O_1},\dots,x_{O_l}}(B(x_{I_1},\dots,x_{I_k})) = \prod_{j=1}^{l} \frac{1}{u_{O_j} - l_{O_j}} \cdot \int_{B(x_{I_1},\dots,x_{I_k})} x_{O_1} \dots\, \mathrm{d}x_{O_l} \tag{6}$$

where the integral is simply *the volume of the region* $B(x_{I_1}, \dots, x_{I_k})$ and it will be denoted $V(x_{I_1}, \dots, x_{I_k}) \doteq Vol(B(x_{I_1}, \dots, x_{I_k}))$. The expression follows the classical idea of probability theory, the probability is the *possible* volume divided by *total* volume. Note that if there are no inputs, the expression is the same as obtained in [14].

The inputs add parameters to the probability therefore it is reasonable to ask when those input actions should be taken in order to *maximize the probability of the test case*. Put in other words, find which $x_{I_1}, \dots, x_{I_k}$ give the maximum volume of $B(x_{I_1}, \dots, x_{I_k})$. But first we focus on practical aspects about the computation of the V function.

VI. Parametric volume computation and evaluation

Given the test case, there is a set $\alpha_1 \dots \alpha_n \alpha_{n+1} \dots \alpha_m$ of participating transitions, which in turn generate a set of inequalities through (2), (3) and (4). For the sake of volume computation it is the same if we take all inequalities as $\leq$. In general these equations form a system of inequalities $A\vec{x} \leq \vec{b}$, where the vector $\vec{x}$ denote the outputs $x_{O_1}, \dots, x_{O_l}$ and inputs $x_{I_1}, \dots, x_{I_k}$, A is a matrix of size $o \times m$ with elements in $\{-1, 0, 1\}$, where o is the number of simple inequalities of the form $\pm x_i \leq c$ or $x_i - x_j \leq c$ generated by previous equations, and $\vec{b}$ is a column vector of integer constants of size m. This kind of system is known as *separation theory* [23], [21], and the model checking community has developed a practical representation called DBM [8].

Then $B(x_{I_1}, \dots, x_{I_k})$ is simply the set of solutions of the system $\{\vec{x} \mid A\vec{x} \leq \vec{b}\}$, therefore we need to compute the volume of a *parametric convex polytope* of dimension l.

The procedure is based on the Fourier-Motzkin (FM) elimination method described in [25]. The same has been done in [14], but our case adds k columns for the inputs.

The FM method *projects* a system of inequalities in a given coordinate in order to eliminate that variable. It is to systems of inequalities what Gaussian elimination is to systems of equalities. It is based on scaling, rearranging and adding of rows, in order to obtain a system with the same set of solutions but more suited to get the lower and upper bounds of the solution set for each coordinate. It is worth mentioning that all operations applied to the system maintain the invariance of being DBM-like. The main difference with Gaussian elimination is that rows are added to the matrix, in a possibly exponential number. (There are many practical details about the FM method, such as unsatisfiability of the system, for this we suggest to read [25], [26].).

After applying the FM method to $B(x_{I_1}, \dots, x_{I_k})$, we obtain a set of matrices, which partition $B(x_{I_1}, \dots, x_{I_k})$ into subregions. In each subregion, the lower and upper bounds of x_{O_i} $(1 \leq i \leq l)$ are the functions of $x_{O_1}, \dots, x_{O_{i-1}}, x_{I_1}, \dots, x_{I_k}$, and the lower and upper bounds of x_{I_j} $(1 \leq j \leq k)$ are the functions of $x_{I_1}, \dots, x_{I_{j-1}}$. The general form of a matrix is shown in Fig. 1.

$$\begin{array}{rl}
x_{O_l} + a^1_2 x_{O_{l-1}} + \dots + a^1_l x_{O_1} + a^1_{l+1} x_{I_k} + \dots + a^1_{l+k} x_{I_1} \leq D^1 \\
-x_{O_l} + a^2_2 x_{O_{l-1}} + \dots + a^2_l x_{O_1} + a^2_{l+1} x_{I_k} + \dots + a^2_{l+k} x_{I_1} \leq D^2 \\
x_{O_{l-1}} + \dots + a^3_l x_{O_1} + a^3_{l+1} x_{I_k} + \dots + a^3_{l+k} x_{I_1} \leq D^3 \\
-x_{O_{l-1}} + \dots + a^4_l x_{O_1} + a^4_{l+1} x_{I_k} + \dots + a^4_{l+k} x_{I_1} \leq D^4 \\
\ddots \qquad\qquad \vdots \\
x_{O_1} + a^{2l-1}_{l+1} x_{I_k} + \dots + a^{2l-1}_{l+k} x_{I_1} \leq D^{2l-1} \\
-x_{O_1} + a^{2l}_{l+1} x_{I_k} + \dots + a^{2l}_{l+k} x_{I_1} \leq D^{2l} \\
x_{I_k} + \dots + a^{2l+1}_{l+k} x_{I_1} \leq D^{2l+1} \\
-x_{I_k} + \dots + a^{2l+2}_{l+k} x_{I_1} \leq D^{2l+2} \\
\ddots \qquad \vdots \\
x_{I_1} \leq D^{2l+2k-1} \\
-x_{I_1} \leq D^{2l+2k}
\end{array}$$

Fig. 1. A matrix generated by the FM method.

From the first $2l$ rows we obtain the affine limits of integration for the outputs $x_{O_1}, \dots, x_{O_l}$, while the rest $2k$ rows constitute the *applicability condition* of the integral based on the first $2l$ rows. The integral can be solved recursively by symbolic computation using the fundamental theorem of calculus $\int_a^b x^k\, \mathrm{d}x = \frac{x^{k+1}}{k+1}\Big|_a^b$, rendering a polynomial of degree l on the k input variables. The sum of all polynomials that have the same applicability function defines the *evaluation function* of the applicability function. The whole set of matrices out of the FM method gives $V(x_{I_1}, \dots, x_{I_k})$ as a piecewise defined continuous function of those polynomials given each applicability condition.

Example: For the previous example STIOA with two inputs, four outputs and test case $\sigma = aijbc$, the set of inequalities generates a volume function which is a piecewise continuous polynomial on x_i, x_j of degree 4 divided in 12 regions whose function V is shown in Fig. 2 and depicted in Fig. 3. □

$$
V(x_i, x_j) = \begin{cases}
-\frac{1}{6}(-5+x_i)^3 & (3 \le x_i < 4 \wedge x_j - x_i < 2 \wedge 4 \le x_j) \vee \\
 & (4 \le x_i < 5 \wedge 0 < x_j - x_i < 2) \\
\frac{1}{3}(49 - 24x_i + 3x_i^2) & 2 < x_i < 3 \wedge 2 < x_j - x_i \wedge 4 \le x_j \\
\frac{1}{2}(6 + x_i - x_j)(-7 + x_j)^2 & 2 < x_i < 3 \wedge 4 < x_j - x_i \wedge x_j < 7 \\
\frac{1}{2}(-1 + x_i)(6 + x_i - x_j)(-7 + x_j)^2 & 1 < x_i < 2 \wedge 4 < x_j - x_i \wedge x_j < 7 \\
-\frac{1}{6}(-1 + x_i - x_j)^2(-13 + x_i + 2x_j) & (2 < x_i < 3 \wedge 0 \le x_j - x_i < 1) \vee \\
 & (3 \le x_i < 4 \wedge 0 < x_j - x_i \wedge x_j < 4) \\
-\frac{1}{6}(-1 + x_i)(-1 + x_i - x_j)^2(-13 + x_i + 2x_j) & 1 < x_i < 2 \wedge 0 \le x_j - x_i < 1 \\
\frac{1}{3}(1 - 24x_i + 3x_i^2 + 24x_j - 3x_j^2) & 2 < x_i < 3 \wedge 1 < x_j - x_i \wedge x_j < 4 \\
\frac{1}{3}(-1 + x_i)(1 - 24x_i + 3x_i^2 + 24x_j - 3x_j^2) & 1 < x_i < 2 \wedge 1 \le x_j - x_i < 2 \\
\frac{1}{6}(49 - 147x_i + 84x_j + 42x_i x_j - 27x_j^2 - 3x_i x_j^2 + 2x_j^3) & 3 \le x_i < 5 \wedge 2 < x_j - x_i \wedge x_j < 7 \\
\frac{1}{6}(22 - 120x_i - 9x_i^2 + x_i^3 + 84x_j + 42x_i x_j - 27x_j^2 - & \\
\quad 3x_i x_j^2 + 2x_j^3) & 2 < x_i < 3 \wedge 2 < x_j - x_i < 4 \\
\frac{1}{6}(74 + 46x_i - 111x_i^2 - 10x_i^3 + x_i^4 - 132x_j + 90x_i x_j + & \\
\quad 42x_i^2 x_j + 33x_j^2 - 30x_i x_j^2 - 3x_i^2 x_j^2 - 2x_j^3 + 2x_i x_j^3) & 1 < x_i < 2 \wedge 2 < x_j - x_i \wedge x_j < 4 \\
\frac{1}{6}(-22 + 142x_i - 111x_i^2 - 10x_i^3 + x_i^4 - 84x_j + 42x_i x_j + & \\
\quad 42x_i^2 x_j + 27x_j^2 - 24x_i x_j^2 - 3x_i^2 x_j^2 - 2x_j^3 + 2x_i x_j^3) & 1 < x_i < 2 \wedge x_j - x_i < 4 \wedge 4 \le x_j
\end{cases}
$$

Fig. 2. Sectional volume function for the running example.

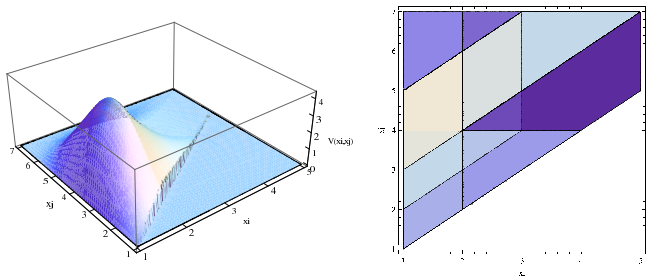


Fig. 3. $V(x_i, x_j)$ and its applicability conditions.

Complexity: The process of building V is divided in three stages: the FM method, the symbolic integration and the generation of the evaluation functions for the applicability conditions.

For a test case ρ without input transitions, the complexity for the first stage, defined on the number of integrals obtained by the FM method, is $O(9^m)$ [14, Section 4.3], where m is the number of transitions involved in ρ. Indeed, if there are k input transitions and l output transition ($k + l = m$) in ρ, the complexity is still $O(9^m)$ because each input transition can be seen as an output transition with the bounds $[0, \infty)$ for the FM method. It is worth noting that for implementation issues there are specific techniques to deal with sparse matrices, like the one in [26], where the authors proposed simplification, variable selection and quasi-syntactic redundancy removal to lessen the exponential growth of matrices.

The complexity for symbolic computation of an integral obtained in the first stage is evaluated on the maximum number of factors generated during integration. Note that we need to integrate on the variables x_{O_i} ($1 \le i \le l$) for output transitions. Moreover, we perform the integration backward, i. e., from x_{O_l} to x_{O_1}. To integrate the j-th variable x_{O_j}, the integration function is of the form

$$\sum C y_1^{r_1} y_2^{r_2} \dots y_p^{r_p} \tag{7}$$

where $y_i \in \{x_{O_1}, \dots, x_{O_j}\} \cup \{x_{I_1}, \dots, x_{I_k}\}$, C is a constant and $q = \sum_{i=1}^{p} r_i \le l - j$. The number of factors in this function is less than that in the following function

$$(y_1 + \dots + y_p + C)^q, \tag{8}$$

which has $O((p+1)^q))$ factors. Due to the important characteristic of the FM method that a bound in the integral involves at most one variable with power 1 and a constant, it is easy to see that $p + q \le m$ and $p \le 2(l - j)$. We also know that $(p+1)^{(q-1)} < p^q$. Therefore, the maximum number of factors in formula (8) is $O((m/2)^{(m/2)})$ if $\frac{m}{2} \le l$ or $O((m-l)^l)$ otherwise.

Obviously, the complexity of computing the evaluation functions for the applicability conditions is linear in the number of integrals obtained in the first stage.

VII. Maximizing the probability of test cases

In (6) the probability of the path was established as a fraction with a numerator that is a function of $x_{I_1}, \dots, x_{I_k}$, the global times when the inputs are activated, and the constant denominator that comes from the system being tested. The maximization problem then has to deal with the numerator, i. e. the integral, and it has special characteristics that make it computationally tractable by a family of numerical algorithms.

The inequalities (2), (3) and (4) form an intersection of convex halfspaces on variables $x_1, \dots, x_m$. Being those variables bounded by system definition, then the inequalities form a convex polytope of dimension m. The integral $\int_{B(x_{I_1},\dots,x_{I_k})} \mathrm{d}x_1 \dots \mathrm{d}x_l$ is then the volume of $B(x_{I_1}, \dots, x_{I_k})$, and it is a function $V : (\mathbb{R}^+)^k \to \mathbb{R}^+$, where $x_{I_1} \dots x_{I_k}$ are the input parameters of the maximization problem.

Let $h(x, i)$ be the hyperplane in $(\mathbb{R}^+)^m$ defined by fixing $x_i = x$, let H be the subspace $H(x_{I_1}, \dots, x_{I_k}) \doteq \{(x_{I_1}, \dots, x_{I_k}, x_{O_1}, \dots, x_{O_l}) \mid x_{O_j} \in \mathbb{R}^+, 1 \le j \le l\}$, that is the intersection of the k hyperplanes $h(x_{I_1}, 1) \cap \dots \cap h(x_{I_k}, k)$ each of them fixing one input of the system, and let K be the convex polytope generated by inequalities (2), (3) and (4). Then the function $V(x_{I_1}, \dots, x_{I_k})$ can also be seen as the volume of the convex polytope $K \cap H(x_{I_1}, \dots, x_{I_k})$ that lays in $(\mathbb{R}^+)^m$, therefore the problem is equivalent to the *maximization of the sectional volume of a convex polytope*.

$$V(x_{I_1}, \dots, x_{I_k}) = Vol(K \cap H(x_{I_1}, \dots, x_{I_k})) \tag{9}$$

In [2] a particular version of this problem was solved, specifically when there is only one parameter for H (therefore it is a hyperplane), that in our STIOA model corresponds to having just one input. Avis et. al. first proved using Brunn-Minkowski inequality [11] that $V(x)$ is *unimodal* and then they used this fact to compute the *exact value* for the maximum using a prune-and-search strategy. Our case is a generalization of this problem, namely instead of having an intersection hyperplane of $m-1$ dimensions, it is a subspace $H(x_{I_1}, \dots, x_{I_k})$ of $m-k < m$ dimensions.

In Fig. 4 there is a convex polytope in $(\mathbb{R}^+)^3$ generated by two outputs and one input satisfying $0 < x_i < x_a < x_b$, $1 \le x_a - x_i \le 3$ and $2 \le x_b \le 5$. There is also a sectional plane in $x_i = 2$. To the right is the continuous function of the sectional area $V(x_i)$. Note the three different parts, being the first a concave quadratic, the second linear and the third convex quadratic.

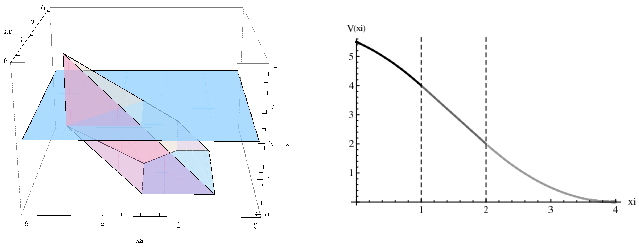


Fig. 4. A convex polytope and its sectional area.

Unimodality of sectional volume: Brunn-Minkowski inequality implies that the function giving the volume of parallel hyperplane sections of a convex body is unimodal [11], [2]. This result is still valid if we extend the sectioning hyperplane to intersections of hyperplanes forming a subspace with parallel borders to the axis. This is exactly what we need for our case.

First we need to extend the concept of unidimensional unimodality [2] to many dimensional functions [11, p.25].

Definition 5: If f is a nonnegative function on $\mathbb{R}^d$, call f *unimodal* if the level sets $L(f,t) \doteq \{x \mid f(x) \geq t\}$ are convex for every $t \geq 0$.

Given a convex polytope $K \subseteq \mathbb{R}^d$ and the subspace generating function $H : \mathbb{R}^c \to \mathbb{R}^d$ with $c < d$, where $H(x_1,\ldots,x_c) \doteq \{(x_1,\ldots,x_c,x_{c+1},\ldots,x_d) \mid x_{c+1},\ldots,x_d \in \mathbb{R}\}$ we define the *sectional volume* of the polytope $V(\vec{x}) \doteq Vol(K \cap H(\vec{x}))$. (Although function H maps into $\mathbb{R}^d$, notice that the result of applying H is a $d-c$ dimensional volume.) We state the Brunn-Minkowski inequality [11] that relates the volume of two convex bodies.

Theorem 1 (Brunn-Minkowski): Given convex polytopes K, L in $\mathbb{R}^d$ and $\lambda \in (0,1)$ the following inequality holds: $Vol(\lambda K + (1-\lambda)L)^{1/d} \geq \lambda Vol(K)^{1/d} + (1-\lambda)Vol(L)^{1/d}$.

The following set inclusion involving the Minkowsky sum is also needed, with $0<\lambda<1$.

$$K \cap H(\lambda_1\vec{x}_1 + (1-\lambda)\vec{x}_2) \supseteq \lambda(K \cap H(\vec{x}_1)) + (1-\lambda)(K \cap H(\vec{x}_2)) \quad (10)$$

The following corollary is the main result of this section.

Corollary 1: The sectional volume $V(\vec{x})$ of a convex polytope is a unimodal function.

Proof: Let $\vec{x_1},\vec{x_2} \in \mathbb{R}^c$ and $S_1 = K \cap H(\vec{x_1}), S_2 = K \cap H(\vec{x_2})$. Clearly S_1, S_2 are convex polytopes in $\mathbb{R}^{d-c}$. Unimodality of $V(\vec{x})$ holds if the level sets $L(V,t)$ are convex for all $t \geq 0$. Now suppose $0<\lambda<1$ and $\vec{x_1},\vec{x_2} \in L(V,t)$, that is $V(\vec{x_1}), V(\vec{x_2}) \geq t$, then we have to show $V(\lambda\vec{x_1}+(1-\lambda)\vec{x_2}) \geq t$.

$$\begin{array}{ll} & V(\lambda\vec{x_1} + (1-\lambda)\vec{x_2})^{1/(d-c)} \\ \geq & \{\ \textit{Using (10)}\ \} \\ & Vol(\lambda S_1 + (1-\lambda)S_2)^{1/(d-c)} \\ \geq & \{\ \textit{Brunn-Minkowski}\ \} \\ & \lambda Vol(S_1)^{1/(d-c)} + (1-\lambda)Vol(S_2)^{1/(d-c)} \\ \geq & \{\ \vec{x_1},\vec{x_2} \in L(V,t)\ \} \\ & t^{1/(d-c)} \end{array}$$

■

Convexity of level sets leads directly to the next proposition.

Proposition 1: For every unimodal function, local maxima coincide with the global maximum.

Maximum Search: As we already said, our problem is a generalization of the one given by [2]. It is worth to remark that this method is not only exact but also fast, since it is linear in the number of faces of the convex polytope in the case of one input variable.

Although to the extent of this work we are not going to extend the aforementioned exact and quick maximum sectional volume computation, the unimodality of V let us *approximate* the maximum value using numerical optimization methods. In general the V function is not differentiable (in a finite set of points) nor it maintains its convexity (see Fig. 4), therefore many classical analytic methods are not applicable. However its unimodality implies that there is one local maximum and it coincides with the global maximum. In the unidimensional case this leads to a family of algorithms that iteratively reduce the *interval of uncertainty* where the maximum is achieved up to an ϵ degree of accuracy [19]. The algorithm is based on an invariant that says the point x^* where f hits the maximum is in the interval $[a,b]$. This interval is narrowed, choosing two internal points $a' < b'$ and comparing their f value. If $f(a') < f(b')$ then the maximum cannot be in $[a,a']$ therefore the leftmost interval is discarded. The case $f(a') \geq f(b')$ is symmetric. When the iterative construction finishes, the middle point $(a+b)/2$ is usually taken as a final approximation of x^*.

{ *Precondition* : $f(x^*) = \max_{x\in[a,b]} f(x)$ }
{ *Invariant* : $x^* \in [a,b]$ }
do $b-a \geq \epsilon \to$
  Choose a', b' *such that* $a<a'<b'<b$
  if $f(a') < f(b') \to a,b := a',b$
  $\square$ $f(a') \geq f(b') \to a,b := a,b'$
  fi
od

The way a', b' are chosen derives in different algorithms, namely, golden ratio search, Fibonacci search, or its variation using Lucas numbers [29]. If the number of evaluations is fixed, Fibonacci search has been proven optimal (in the number of evaluations of the function) for finding the maximum of a unimodal function [13]. For large number of iterations Fibonacci search and golden ratio search coincide. Suppose we choose Fibonacci search, therefore in the long run the bracketing interval decreases $2/(1+\sqrt{5}) \approx 0.61803$ each time. The following rule of thumb is given in [24]: there is no need to go for an ϵ smaller than 10^{-4} or 3×10^{-8} for single or double precision floating point numbers respectively. This implies approximately 20 or 36 iterations for each case, adding 4 iterations per digit of the initial bracketing interval length.

The case for many dimensions is described in [16] and it is based on a recursion in the dimensions of the search space. The base case is covered by the unidimensional algorithm. Suppose we have an algorithm to approximate the maximum for d dimensions $\max_{x_1,\ldots,x_d} f(x_1,\ldots,x_d)$, we can construct

an algorithm for $d+1$ using the unidimensional algorithm over the function $\max_{x_1,\ldots,x_d} f(x_1,\ldots,x_d,\mathbf{x_{d+1}})$. Therefore this algorithm is $O(N^d)$, where N is the number of iterations for the unidimensional case.

A crucial assumption for this class of algorithms is that the region of uncertainty where the maximum lies has to be known beforehand, and the size of this region is an important complexity factor. The following is a procedure to get the initial bracketing for one of the many pieces V can have. Recall that FM method maintains DBM-like inequalities, therefore the applicability condition can be of two forms: $\pm x_i \leq c$ and $x_i - x_j \leq c$. Since we ask the body defined by (2)–(4) to be bounded, so this will be the initial bracketing.

In [23] a system of inequalities is encoded in a weighted directed graph to decide satisfiability. The vertices are the variables $x_1, ..., x_m$ with a special vertex representing 0. For each inequality $x_i - x_j \leq c$ the edge $x_i \xrightarrow{c} x_j$ is added. In the case of $x_i \leq c$, we add edge $x_i \xrightarrow{c} 0$, changing the arrow direction for $-x_i \leq c$. Then the shortest path from x_i to 0 (0 to x_i) is the minimal upper (maximal lower) bound of x_i. Using the Bellman-Ford single source shortest path algorithm, in $O(m^2)$ steps we get all shortest paths from 0 to any x_i, that is all maximal lower bounds. Flipping the direction of all edges and computing again we get the shortest paths from x_i to 0, obtaining the desired minimal upper bounds.

Example: For the running example the maximum volume is 4.333 achieved at $x_i = 2, x_j = 4$. Note that the maximum is obtained pressing i as soon as we are sure a occurred, while j has to be delayed two time units in a trade-off between allowing b to happen first, and avoiding d to occur. □

VIII. On-the-fly probability maximization

Given a system, we have showed how to compute all the input action times in order to maximize the probability of a test case σ. However, once some part of the test case has been executed it is likely that the values originally calculated are not optimal anymore knowing that certain values (the past values) will not change in the future. In this section, we address the question of whether these timings can be recomputed while the test case is executing so that we can improve the probability of success of the test case.

That is, we want to be adaptive along the test case execution based on the *timed history* of such execution. Based on this time history the algorithm should decide which is the optimal time to feed the next input so that the probability of success of the remaining part of the test case is maximized. Therefore we need to define the concept of timed history.

Definition 6: A sequence $\langle \alpha_1, C_1\rangle, \ldots, \langle \alpha_i, C_i\rangle$ is a *timed history* of path $\sigma = \alpha_1, \ldots, \alpha_n$ if $0 < C_1 < \cdots < C_i$ and $i \leq n$.

Given a path and a timed history of it, the system of inequalities has to change in order to reflect that all global time points $x_1, \ldots, x_i$ are fixed according to the timed-history. This involves adding i equalities to the system.

$$x_1 = C_1, \ldots, x_i = C_i$$

Note that inequality (2) implies that every other absolute clock should be greater or equal than the last clock value C_i. The previous modification can lead to nontrivial arrangements in the system of inequalities and this implies, in principle, a complete recomputation of the function V.

Example: Here we have two processes with one input and three outputs, being $a!$ an output transition with a pairing input transition $a?$ that synchronize the two processes.

$$\bullet s_1 \xrightarrow{a?} \bullet s_2 \xrightarrow[[1,2]]{b} \bullet s_3$$

$$\bullet t_1 \xrightarrow[[1,2]]{a!} \bullet t_2 \xrightarrow{i} \bullet t_3 \xrightarrow[[1,2]]{c} \bullet t_4$$

The inequalities for the test case $\sigma = aib$ are:

$$\begin{array}{ll} 0 < x_a < x_i < x_b & 1 \leq x_a \leq 2 \\ x_b < x_c & 1 \leq x_b - x_a \leq 2 \\ & 1 \leq x_c - x_i \leq 2 \end{array}$$

and the volume function is:

$$V(x_i) = \begin{cases} \frac{1}{6}(1 - 6x_i + 6x_i^2 + x_i^3) & 1 < x_i < 2 \\ \frac{1}{6}(-33 + 39x_i - 12x_i^2 + x_i^3) & 2 \leq x_i < 3 \\ \frac{1}{2}(16 - 8x_i + x_i^2) & 3 \leq x_i < 4 \end{cases}$$

$V(x_i)$ achieves a maximum of 0.8987 at $x_i = 2.26795$. If the output $x_a = \frac{3}{2}$ gets fixed in the timed history, the volume function changes in a non-trivial way.

$$\tilde{V}(x_i) = \begin{cases} \frac{1}{8}(-17 + 20x_i - 4x_i^2) & \frac{3}{2} < x_i < \frac{5}{2} \\ \frac{1}{2}(7 - 2x_i) & \frac{5}{2} \leq x_i < \frac{7}{2} \end{cases}$$

Now the maximum is 1 at $x_i = 2.5$, showing that the probability could increase given that we have more information (The intervals have length 1, therefore the probability is exactly the volume.). Also note that the x_i corresponding to previous V is no longer optimal, $\tilde{V}(2.26795) = 0.973$. □

It is worth mentioning that there are specific techniques to deal with the projections of a system that is mix of equalities and inequalities, where the Gaussian elimination and the FM method are interleaved. Moreover in [26] it is argued that adding equalities $x = C$ instead of the equivalent two opposite inequalities $C \leq x \leq C$ is an opportunity to lessen the matrix growth in the projection process.

Suppose the system is evolving, namely the timed history is growing in length. There are two possible cases for the next incoming event, it could be output or input.

For the output case, there is nothing to do except recording its occurrence time. Note that it is possible that given this information the probability of the test case decreases. For example, given a test case generating the area $x_1 \leq x_2$ in the square $[0, 1]^2$, even though the surface occupies half the square (a fair coin), if x_1 falls in the second half, the probability of remaining in the test case would be $1 - x_1$, becoming unfair.

For the input case the situation is different since we can effectively act, taking into account the current timed history and achieve the maximum probability for the recorded information. Given a running system and its timed history $\langle \alpha_1, C_1\rangle, \ldots, \langle \alpha_i, C_i\rangle$, where $\alpha_{i+1} = \alpha_{I_j} \in \Sigma_I$ is the next incoming input, we add the i equalities to the system, V is

computed to reflect those changes, and later it is maximized for all the remaining input variables. Finally this vector is projected on the first coordinate x_{i+1}, giving the exact time when to feed the input.

It is worth noting that only $x_{i+1} = x_{I_j}$ is used and the rest of the maximizing values $x_{I_{j+1}}, \ldots, x_{I_k}$ are discarded. Being the time complexity of the maximization process exponential in the number of inputs, discarding $k-j-1$ values seems a waste of resources. One possibility would be to maximize only in x_{I_j}, but in Fig. 3 we can see that $V(x_i, x_j)$ cannot be maximized fixing some arbitrary value of x_j. Unfortunately, there is no reason to save the costly computed values $x_{I_{j+1}}, \ldots, x_{I_k}$, since, as we have already seen, recording one occurring output and adding it as an equality to the inequalities system, leads to a different sectional volume function $\tilde{V}$. Previous values $x_{I_{j+1}}, \ldots, x_{I_k}$ could only be suboptimal for $\tilde{V}$ maximization, since it is also unimodal and therefore has only one maximum.

The race against the execution time of the test case: All computations to get the optimal input times are costly and in general not compatible with the real-time nature of the system. While computing on-the-fly the next input time $x_{i+1} = x_{I_j}$, it can be quite possible that computation finishes *after* the calculated optimal value, which may render the test useless. In this sense, the computation time races against the wall-clock time in which events occur, and hence it makes necessary to give up calculations in favour of *the best* possible probability (rather than the maximum).

In the following, we propose an algorithm that takes into account the global clock and decides whether it is convenient to stop computations and, in such a case, if it is better to keep the currently approximated value or resort to previously computed values. The algorithm has two phases: the computation V and the maximization V. Suppose the system is running, the timed history $\langle \alpha_1, C_1 \rangle, \ldots, \langle \alpha_i, C_i \rangle$ has been fixed, and $\alpha_{i+1} = \alpha_{I_j}$ is the next input to be fed in the test case, with precomputed value x_{I_j}. Note that there is always a precomputed value for this input since previous to the test case execution, we compute the best values $x_{I_1}, \ldots x_{I_k}$ for input times. Besides, later in the execution, the remaining input values may be recomputed in each new input time choice. For the first phase, the timeout is x_{I_j}, because we cannot be sure that in case we finish the FM method and the symbolic integral computation, the maximization would render a value that is not only better but also ahead of x_{I_j}. If the global clock reaches x_{I_j} while computing the volume function, the computation is discarded and transition α_{i+1} is taken immediately. In the second phase the calculation of the new x_{I_j} takes place. As the calculation is based on an approximation algorithm that iteratively narrows the interval $[a, b]$ in which the optimal reside, we have more information to decide the choice of x_{I_j}. So, in a sense, we have to define an algorithm that compute its own deadline. In the following we give a proposal for such algorithm.

Notice that there are two time values involved: one is the global clock value $x = cl(gt)$, and the other one, the value that is currently being calculated in the approximation interval $[a, b]$. We take $\tilde{x}_{I_j} = \frac{a+b}{2}$ to be best approximation for this last value. (Notice that the precomputed activation time x_{I_j} is of no use now since we already computed the new volume V and we are in a position to compute a value that is proper for this modified volume function.) Two orders between $x, \tilde{x}_{I_j}$ are possible. If $\tilde{x}_{I_j} < x$, we assume that the best time has already occurred and we feed input α_{I_j} immediately. Otherwise $x < \tilde{x}_{I_j}$, in which case we assume that there is still time to continue the computation of the optimal value. If the computation finishes, this will define new optimal values for the remaining input times; in particular, the new value $x_{I_j} > x$ which is the optimal time value to feed the current input.

With this procedure the loop guard on the maximum search algorithm can be independent of any ϵ, and rest on the variant function $\tilde{x}_{I_j} - x$ given the above real-time conditions. Stopping the loop to wait for the clock to reach $\tilde{x}_{I_j}$ or continue approximating the best value, is an implementation decision later to be taken in the design process.

Irrespective of the way of exiting this loop, we always store the values for all the remaining input variables, in order to use them in the first phase of next incoming inputs.

IX. Conclusions

In this paper we extended [14] adding inputs to the stochastic timed transition system. In this augmented model we showed that the problem of computing the best input times in order to maximize the probability of a test case execution, is equivalent to maximizing the sectional volume of a convex polytope when the probability distributions involved are uniform. The sectional volume function was computed using the FM method obtaining a piecewise continuous map. This function was proven to be unimodal extending a known result [11], [2]. The maximum was approximated to a desired degree of accuracy using numerical optimization methods specifically tailored to unimodal functions. Then we showed how this maximum computation could be adaptive to a running system where more information is available as a timed history.

We also showed that the complexity of most operations involved in the whole process (the FM method, the symbolic integration, function evaluation and maximization) are basically exponential in the path length. Although this seems contrary to real-time, we presented two strategies to avoid that problem. First, we showed how to compute the best values before the system starts running when all the available information are the causality relation of transitions, and the interval of time when the outputs could occur. Then, in the case we want to do it on-the-fly and profit from the time information recollected, we presented a partial computation method that does the best effort against a racing output transition that is likely to occur first. In [9] it is stated that the exact volume computation of convex polytopes is *#P-hard* (it contains the *NP* class complexity) which is precisely our problem: we deal with a function that is the (sectional) volume of a convex polytope of dimension given by the number of inputs.

There are opportunities for specific optimizations that could help to lessen this time complexity in practice. For example,

there are techniques for sparse systems [26], but, to the authors knowledge, there is no specific optimizations of FM method tailored to DBM-like inequality systems like the ones generated by our test cases. (Research areas like Timed Automata Model Checking, Constraint Logic Programming and Abstract Interpretation could also profit from that algorithm.) Also in [26] there are techniques that trade precision for improved tractability in the FM method, avoiding the exponential growth of the matrix in the first phase of our computation. Since this is compatible with later phases (e.g. Fibonacci search), we could also profit from this idea. Another possible source of improvement lies on the integration procedure. We have shown that the classical way to compute integration, which is used in Section VI, has a high complexity, and therefore it is time consuming. In [3] a system of integral equations is converted into an equivalent system of differential equations to allow fast computation. A challenge in applying this approach to our case is how to handle parameters during converting integral equations and solving differential equations. There are also probabilistic approaches to volume estimation giving super-polynomial speed-up [15], obtaining randomized polynomial algorithms in the dimensions of the convex polytope.

In case of the on-the-fly probability maximization, we showed an example where adding a point in the timed history as an equality changes the shape of the volume function in a way that is not clear if previous computations can be reused. We also found models and test cases where that single equality only produces a selection of pieces of the original sectional volume function or a minor variation of it, instead a completely new function. Those cases could be detected in order to reuse previous computations. In addition, the dynamics of the race between output transitions and the next input time under calculation has not been explored. For instance, it could be the case that in the approximation interval $[a, b]$ during maximum search b stays fixed and it is only a that grows. If such speed is growing faster than the wall clock, it may be reasonable to continue the computation rather than aborting.

In [2] the problem of maximum sectional volume of a convex polytope is solved for hyperplanes, that in our setting means having just one input transition. Moreover this solution is not an approximation. Achieving a generalization of this Computational Geometry algorithm would render a symbolic computation of the maximum, where the next incoming input feed time is a function of the timed history. The obvious question is whether this approach could lead to relief the now expensive on-the-fly computation to a simple function evaluation, moving most of the work to off-line processing.

It would also be interesting to relate this work to cost optimal reachability analysis for timed automata using priced zones [17]. Although probabilities are not considered per se, they could be encoded as prices and Uppaal CORA could generate a trace that is globally optimal with respect to price.

Acknowledgements: We would like to thank Peter Niebert for fruitful discussions and also to the anonymous referees for their careful reading and references to related work.

References

[1] R.B. Ash and C.A. Doléans-Dade. *Probability & Measure Theory*. Academic Press, 2000.

[2] D. Avis, P. Bose, G.T. Toussaint, T.C. Shermer, B. Zhu, and J. Snoeyink. On the sectional area of convex polytopes. In *Symposium on Computational Geometry*, pages C–11–C–12, 1996.

[3] M. Bernadsky and R. Alur. Symbolic analysis for GSMP models with one stateful clock. In *Procs. HSCC 2007*, volume 4416 of *LNCS*, pages 90–103. Springer, 2007.

[4] L. Brandán Briones and E. Brinksma. A test generation framework for *quiescent* real-time systems. In *Procs. of FATES 2004*, volume 3395 of *LNCS*, pages 64–78. Springer, 2005.

[5] L. Brandán Briones and M. Röhl. Test derivation from timed automata. In Broy et al. [6], pages 201–231.

[6] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors. *Model-Based Testing of Reactive Systems, Advanced Lectures*, volume 3472 of *LNCS*. Springer, 2005.

[7] D. Clarke and I. Lee. Automatic test generation for the analysis of a real-time system: Case stud. In *Procs. of RTAS '97*, pages 112–124. IEEE, 1997.

[8] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Procs. of the International Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 197–212. Springer, 1990.

[9] M.E. Dyer and A.M. Frieze. On the complexity of computing the volume of a polyhedron. *SICOMP: SIAM Journal on Computing*, 17, 1988.

[10] M.A. Fecko, M. Ümit Uyar, A.Y. Duale, and P.D. Amer. A technique to generate feasible tests for communications systems with multiple timers. *IEEE/ACM Trans. Netw.*, 11(5):796–809, 2003.

[11] R.J. Gardner. The Brunn-Minkowski inequality. *Bulletin of the American Mathematical Society*, 39(3):355–405, 2002.

[12] P.W. Glynn. A GSMP formalism for discrete event simulation. *Proceedings of the IEEE*, 77(1):14–23, 1989.

[13] S.M. Johnson. Best exploration for maximum is fibonaccian. Technical Report P-856, Rand Corp., 1956.

[14] M. Jurdziński, D. Peled, and H. Qu. Calculating probabilities of real-time test cases. In *Procs. FATES 2005*, volume 3997 of *LNCS*, pages 134–151. Springer, 2006.

[15] R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $O*(n^5)$ volume algorithm for convex bodies. *RSA: Random Structures & Algorithms*, 11, 1997.

[16] P.D. Krolak and L. Cooper. An extension of fibonaccian search to several variables. *Commun. ACM*, 6(10):639–641, 1963.

[17] K.G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T.S. Hune, P. Petterson, and J.M.T. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Procs. CAV'01*, 2001.

[18] G. Luo, G. von Bochmann, and A. Petrenko. Test selection based on communicating nondeterministic finite-state machines using a generalized wp-method. *IEEE Trans. Software Eng.*, 20(2):149–162, 1994.

[19] J.M. Martínez and S.A. Santos. *Métodos Computacionais de Otimização*. Colóquio Brasileiro de Matemática, Apostilas. 20. Sociedade Brasileira de Matemática, 1995.

[20] M.G. Merayo, M. Núñez, and I. Rodríguez. Generation of optimal finite test suites for timed systems. In *TASE*, pages 149–158. IEEE, 2007.

[21] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.

[22] B. Nielsen and A. Skou. Automated test generation from timed automata. *STTT*, 5(1):59–77, 2003.

[23] V.R. Pratt. Two easy theories whose combination is hard. Technical report, November 13 1977.

[24] W.H. Press et al. *Numerical Recipes in C (Second Edition)*. 1992.

[25] M. Schechter. Integration over a polyhedron: an application of the Fourier-Motzkin elimination method. *The American Mathematical Monthly*, 105(3):246–251, 1998.

[26] A. Simon and A. King. Exploiting sparsity in polyhedral analysis. In *Procs. SAS 2005*, volume 3672 of *LNCS*, pages 336–351. Springer, 2005.

[27] J. Springintveld, F.W. Vaandrager, and P.R. D'Argenio. Testing timed automata. *Theor. Comput. Sci.*, 254(1-2):225–257, 2001.

[28] J. Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.

[29] B. Yildiz and E. Karaduman. On Fibonacci search method with k-Lucas numbers. *App. Math. and Comp.*, 143(2–3):523–531, 2003.