ISSN 2007-9737

Axiomatization, Computability and Stability for Discrete Event Time Algorithms

Zvi Retchkiman Königsberg

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

mzvi@cic.ipn.mx

Abstract. This work proposes a formalization of discrete event time algorithms. A Church type thesis, its proof, and the notion of stability for discrete event time algorithms are presented. The Church thesis for discrete algorithms motivates us to consider the Church thesis for the case when we are dealing with discrete event time algorithms. The notions of discrete event time algorithm and discrete event time dynamical system are postulated to be equivalent. The stability concept for discrete event time algorithms is defined. The stability analysis presentation starts concentrating in discrete event time algorithms, i.e., discrete event time dynamical systems, whose Petri net model is described by difference equations, and continues considering Lyapunov energy functions in terms of the logical structures of the vocabulary. A stability analysis based on the reachability tree of the Petri net model is also discussed.

Keywords. Discrete event time algorithms, discrete event time dynamical systems, Church thesis, Petri nets, stability.

1 Introduction

In this paper the notion of discrete time event algorithms is introduced. The paper starts proposing a formalization, free of any interpretation, of discrete event time algorithms, and continues discussing computability issues. A Church type thesis, its proof, and the notion of stability for discrete event time algorithms are proposed.

The Church thesis for discrete algorithms motivates us to consider the Church thesis for the case when we are dealing with discrete event time algorithms. Gurevich [1] has shown that any algorithm that satisfies three postulates can be step-by-step emulated by an abstract state machine (ASM). Adding two more postulates, Dershowitz and Gurevich [2] proceeded to prove that all notions of algorithms for common discrete-time models of computation in computer science are covered by this formalization.

This includes Turing machines, Minsky counter machines, Post machines, random access memory (RAM) and so on. Bournez, Dershowitz and Neron [3] have formalized a generic notion of analog algorithm, their proposed framework is an extension of [1, 2]. They provide postulates defining analog algorithms in the spirit of those given for discrete algorithms, and continue proving some completeness results. Our presentation follows a similar construction to the one given for analog algorithms.

The notions of discrete event time algorithm and discrete event time dynamical system are postulated to be equivalent. The stability concept for discrete event time algorithms is defined. The stability analysis presentation starts concentrating in discrete event time algorithms i.e., discrete event time dynamical systems, whose Petri net model is described by difference equations, and continues considering Lyapunov energy functions in terms of the logical structures of the vocabulary.

A stability analysis based on the reachability tree of the Petri net model is also discussed. The paper is organized as follows. In section 2, axiomatization and computability issues related to discrete event time algorithms are presented. Section 3, discusses the stability problem for discrete event time algorithms in terms of its equivalent discrete event time dynamical system. Section 4 presents an application example. Section 5 discusses the states, (equivalently strucures), stability problem for discrete event time algorithms in terms of Lyapunov energy functions, and finally in section 6, a stability analysis based on the reachability tree is also discussed.

2 Discrete Event Time Algorithms, the Church Thesis

Gurevich [1] has shown that any algorithm that satisfies three postulates can be step-by-step emulated by an abstract state machine (ASM). Adding two more postulates, Dershowitz and Gurevich [2] proceeded to prove that all notions of algorithms for common discrete-time models of computation in computer science are covered by this formalization. Bournez, Dershowitz and Neron [3] have formalized a generic notion of analog algorithm, their proposed framework is an extension of [1, 2].

They provide postulates defining analog algorithms in the spirit of those given for discrete algorithms, and continue proving some completeness results. The next presentation follows a similar construction to the one given for analog algorithms, adapting it for discrete event time algorithms. In this section the formalization, free of any interpretation, of the notion of discrete event time algorithm as well as a Church type thesis based on the previous cited work are presented.

The reader looking for a more detailed discussion of the ideas behind the following presentation is encouraged to review [1, 2, 3] but in particular [1] is recommended.

Definition 1 *A dynamical system is a four-tuple $\{T, X, A, \phi_t\}$, where T is called the time set, X is a state space, A is the set of initial states $A \subset X$ and $\phi_t : X \to X$ is a family of evolution operators parameterized by $t \in T$ satisfying the following properties: for $x \in X, \phi_0(x) = x$, and $\phi_{t+s} = \phi_t \circ \phi_s$.*

Remark 2 *Note that in our definition of dynamical system, it is allowed to have, in general, more than one evolution operator.*

When $T = \mathcal{N} = \{0, 1, 2, \cdots\}$ we speak of a discrete event time dynamical system. We will consider T equipped with the absolute value as a normed space i.e., $(T, |\cdot|)$. A discrete event time dynamical system is generally represented by difference equations.

Remark 3 *When dealing with discrete event time dynamical systems determined by difference equations on $\mathcal{R}^n$, we define the euclidean metric d as: $d(x, y) =| x - y |= \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}, \forall x, y \in \mathcal{R}^n$. X equipped with the above euclidean metric defines a metric space.*

Definition 4 *A discrete event time dynamical system is said to be computable if and only if its family of evolution operators (also called its trajectories) is obtained as solutions of its mathematical model.*

Postulate A. A discrete event time algorithm is a discrete event time dynamical system.

Definition 5 *A vocabulary $\mathcal{V}$ is a finite collection of fixed-arity (possibly nullary) function symbols. We assume that $\mathcal{V}$ contains the scalar (nullary) function true. A first-order structure X of vocabulary $\mathcal{V}$ is a non-empty set S, the base set (domain) of X, together with interpretations of the function symbols in $\mathcal{V}$ over S, denoted by $\|f\|_{\mathcal{X}}$. Similarly, the interpretation of a term $f(t_1, \cdots, t_n)$ in X is recursively defined by $\|f(t_1, \cdots, t_n)\|_X = \|f\|_X(\|t_1\|_X, \cdots, \|t_n\|_X)$. Let X and Y be structures of the same vocabulary $\mathcal{V}$. An isomorphism from X onto Y is a one-to-one function ζ from the base set of X onto the base set of Y such that $f(\zeta t_1, \cdots, \zeta t_n)) = \zeta(t_0)$ in Y whenever $f(t_1, \cdots, t_n) = t_0$ in X.*

Definition 6 *A state transition system, consists of a set of states S, a subset I of initial states, transition functions on states, which determines the next-state relation, (states with no "next" state, will be terminal states).*

Postulate B (Abstract state). States are first order structures with equality, sharing the same fixed, finite vocabulary. States and initial states are closed under isomorphism. Transitions preserve the domain, and transitions and isomorphisms commute.

Remark 7 *In more simple words, states are structures which are invariant under isomorphism, and as a consequence they result to be very general and therefore any model of computation results to be a special case of this. The transitions are governed by a finite description.*

Definition 8 *An abstract transition system is a state transition system, whose function symbols f are interpreted as functions, and that satisfies postulate B, where the transition function on states is equal to ϕ_t.*

Postulate C. A discrete event time algorithm is an abstract transition system.

Definition 9 *If f is a j-ary function symbol in vocabulary $\mathcal{V}$, and a is a j-tuple of elements of the base set of X, then the pair (f, a) (also denoted by $f(a)$) is called a location. We denote by $\|f(a)\|_X$ its interpretation in X. If (f, a) is a location of X then $(f, a, \|f(a)\|_X)$ is an update of X. When Y and X are structures over the same domain and vocabulary, $Y \setminus X$ denotes the set of updates $\triangle^+ = \{(f, a, \|f(a)\|_Y) : \|f(a)\|_Y \neq \|f(a)\|_X|\}$.*

Definition 10 *An infinitesimal generator is a function $\triangle$ that maps the state space X to a set $\triangle(X)$ of updates, and preserves isomorphisms i.e., if ζ is an isomorphism of states X, Y, then for all updates $(f, a, \|f(a)\|_X) \in \triangle(X)$, we have an isomorphic update $(f, \zeta a, \zeta\|f(a)\|_X) \in \triangle(Y)$.*

Definition 11 *A semantics ψ over a class C of sets S is a partial function mapping initial evolutions over some $S \in C$ to an element of S.*

Definition 12 *The infinitesimal generator associated with a semantics ψ maps the state space X, such that $\psi[X, f, a] = \psi(\|f(a)\|_{\phi_t(X)})$ is defined for all locations (f, a), to the set of updates $\triangle_\psi(X) = \{(f, a, \psi[X, f, a]) : (f, a) \in X\}$*

Remark 13 *When $T = \mathcal{N}$, an example of semantics over the class of all sets would be the function $\psi_\mathcal{N}$ mapping f to $\psi_\mathcal{N}(f_n) = f_{n+1}, n \in \mathcal{N}$.*

Remark 14 *From now on, we fixed the semantics ψ to be equal to the function $\psi_\mathcal{N}$ mapping f to $\psi_\mathcal{N}(f_n) = f_{n+1}, n \in \mathcal{N}$. However,it is assumed that the class of discrete event time dynamical systems is restricted to those that guarantee the existence of the respective semantics and as a result its associated set of updates is well defined. Therefore, not all possible discrete event time dynamical systems are allowed.*

Postulate D. For any discrete event time algorithm, there exists a finite set $Terms$ of variable free terms over the vocabulary $\mathcal{V}$, such that for all states X and Y that coincide for $Terms$, $\triangle_\psi(X) = \triangle_\psi(Y)$.

An abstract state machine, or ASM, is a state-transition system in which algebraic states (no predicate symbols) store the values of functions of the current state. Transitions update a finite number current states and are programmed using a convenient language based on guarded commands for updating individual states. ASM captures the notion that each step of an algorithm performs a bounded amount of work, whatever domain it operates over, so are central to the development.

Definition 15 *[1] An abstract state machine (ASM) is given by: a set S of algebraic states (no predicate symbols), closed under isomorphism, sharing a vocabulary $\mathcal{V}$, a set (or proper class) I of initial states, closed under isomorphism, and a program P, consisting of finitely many commands, each taking the form of a guarded assignment:*

$$if \quad q \quad then \quad t := u,$$

for terms t and u over the vocabulary, and q is a conjunction of equalities and inequalities between *terms i.e., given a state α which belongs to S, program P defines and therefore computes the following sub-set of the set of updates $\triangle^+$, $\{f(\|s\|_\alpha) := \|u\|_\alpha : (if \quad q \quad then \quad f(s) := u) \quad \in P \quad and \quad \|q\|_\alpha = \quad true\}$.*

In addition to the rule of the ASM program (see definition 15), we introduce the following rules.

Definition 16 *If each $R_1, R_2, \cdots, R_k$ are rules of the ASM i.e.:*

$$if \quad q \quad then \quad t := u,$$

then:

$$par \quad R_1, R_2, \cdots, R_k \quad endpar,$$

is a rule which executes them in parallel, with $\triangle_\psi(X)$ equal to the union of the same sub-set of updates given in definition (15) for each $R_1, R_2, \cdots, R_k$.

Definition 17 *A rule of the form $Dynamic(f(t_1, t_2, \cdots, t_j), t_0)$ where f is a symbol of arity- j and, $t_0, t_1, t_2, \cdots, t_j$ are variable free terms, then the rule is defined by $\psi[X, f, (t_1, t_2, \cdots, t_j)] = \psi(f(t_1, t_2, \cdots, t_j)) := t_0$, where $\{\psi[X, f, (t_1, t_2, \cdots, t_j)]\}$ is an element of the set of updates $\triangle_\psi(X)$. In addition if $R_1, R_2, \cdots, R_k$ are rules of the form $Dynamic$ then:*

$$par \quad R_1, R_2, \cdots, R_k \quad endpar,$$

is also a rule, with $\triangle_\psi(X)$ being equal to the union of $\{(f_i, a_i, \psi[X, f_i, a_i]) : (f_i, a_i) \in X\}$ for $i = 1, \cdots, k$.

Definition 18 *If ϕ is a Boolean term and R_1 and R_2 are rules then, if ϕ then R_1 else R_2 is a rule.*

The following result plays a fundamental role in the proof of the Church thesis for discrete event time algorithms [3].

Theorem 19 *For every algorithm of vocabulary $\mathcal{V}$, there is an ASM program, which for all states has the identical set of updates i.e., emulates it step by step.*

Example 20 *[6] Let us consider the matrix difference equation describing the discrete event time dynamical behavior of a Petri net (PN) with m places and t transitions represented as [5]:*

$$M_{n+1} = M_n + A^T u_n, n \in \mathcal{N}, M_n \in \mathcal{N}^m \\ \text{and } u_n \in \{0,1\}^t. \quad (1)$$

This evolution is described by its associated set of updates of the following program rule:

$$\begin{gathered} par \\ \text{Dynamic}\left(M_n(p_1), M_n(p_1) + \sum_{j=1}^{t} a_{j1} u_n(j)\right), \\ \cdots, \qquad (2) \\ \text{Dynamic}\left(M_n(p_m), M_n(p_m) + \sum_{j=1}^{t} a_{jm} u_n(j)\right) \\ endpar. \end{gathered}$$

Notice that if M' can be reached from some other marking $M = M_n$ for some $n \in \mathcal{N}$ through a firing sequence $\{u_0, u_1, ..., u_{d-1}\}$ writing equation (1) for each one of the elements of the firing sequence, and summing up, we obtain that:

$$M' = M + A^T u, \; u = \sum_{k=0}^{d-1} u_k. \quad (3)$$

Equation (3) would result in an ASM program, where the program rule (2) appears d times.

Definition 21 *A $\psi - ASM$ comprises the following: an ASM program, a set S of first-order structures with equality over some finite vocabulary $\mathcal{V}$ closed under isomorphisms with a subset S_0 of S closed under isomorphisms, and a well defined update set of computations $\triangle_\psi$ associated with ψ.*

Definition 22 *A discrete event time algorithm is a $\psi - ASM$ which satisfies postulates A, B, C and D.*

We are assuming for that for each discrete event time algorithm, the trajectories can be computed from the description of its discrete event time dynamical system. In other words not all discrete event time algorithms are contemplated just those of them which guarantee their existence.

Definition 23 *A semantics ψ is unambiguous if for all sets S of first-order structures over some finite vocabulary $\mathcal{V}$ closed under isomorphisms, and for all subsets $S_0 \in S$ closed under isomorphisms, whenever there exists some ϕ and a $\psi - ASM$, then ϕ is unique.*

Theorem 24 *Assuming ψ is unambiguous, for every process (algorithm) satisfying the postulates A, B, C and D, there is an equivalent $\psi - ASM$.*

Remark 25 *In simple words it emulates any algorithm that satisfy the postulates.*

Theorem 26 *(The Church thesis for discrete event time algorithms) The discrete event time algorithm is computable if and only if the $\psi - ASM$ computes its equivalent discrete event time dynamical system.*

Proof. By hypothesis the discrete event time algorithm is computable which implies that its equivalent discrete event time dynamical system is computable (recall definition 4). Therefore, there exists a procedure (algorithm) which computes its trajectories from its mathematical model description and as a consequence, the $\psi - ASM$ program will be able to emulate the algorithm step by step and therefore compute these trajectories by a proper definition of its rules (see 19, 21). For the other side of the implication, given a $\psi - ASM$ program which first interprets the fixed discrete event time dynamical system and then computes its trajectories, we define a numerical procedure which mimics it and therefore computes the discrete event time dynamical system's trajectories. ■

3 Stability of Discrete Event Time Algorithms

In this section, we will focus our attention to study the class of discrete event time dynamical systems whose Petri net model is described by difference equations, leaving the reachability tree analysis technique for section 6. We will begin by recalling some basic definitions in stability theory for this class [6, 8].

Definition 27 *(Stability)*
Let us consider a discrete event time dynamical system represented by the following difference equation:

$$x(n+1) = f[n, x(n)] : x(n_o) = x_0,$$
$$n \in \mathcal{N}_{n_0}, x(n) \in \mathcal{R}^n,$$
$$f : N_{n_0} \times \mathcal{R}^n \to \mathcal{R}^n \text{continuous} \qquad (4)$$

We say that state $x = 0$ of system (4) is stable if and only if, $\forall n_0 \in \mathcal{N}$ and $\forall \varepsilon > 0 \; \exists \; \delta = \delta(n_0, \varepsilon) > 0$ such that if $\| x_0 \| < \delta \Rightarrow \| x(n, n_0, x_0) \| < \varepsilon \; \forall n \in N^+_{n_0}$.

Now, let us divide the set of structures i.e., the set of states of the discrete event time dynamical system, in unstable and stable sets $X = \{X_{un}, X_s\}$.

Definition 28 *A discrete event time algorithm is said to be stable if and only if the discrete event time dynamical system is stable. A discrete event time dynamical system is stable if and only if the unstable structures are empty or they are not attained as the program of the $\psi - ASM$ executes.*

Let us suppose that it is possible to pass from unstable structures to stable structures by properly defining the rules of the $\psi - ASM$ program, then we will obtain a stable discrete event time algorithm i.e., we have managed to stabilize the unstable algorithm i.e., the discrete event time dynamical system is stabilizable.

Definition 29 *A discrete event time algorithm is said to be stabilizable if and only if it is possible to avoid the unstable structures by properly defining the rules of the $\psi - ASM$ program.*

Let us consider the matrix difference equation describing the discrete event time dynamical behavior of a Petri net with m places and t transitions represented as [5]:

$$M' = M + A^T u, \; u = \sum_{k=0}^{d-1} u_k. \qquad (5)$$

Pick as our Lyapunov function candidate $v(M) = M^T \Phi$ with Φ an m vector (to be chosen). Then we can state and prove the following result [6].

ISSN 2007-9737

Proposition 30 *Let PN be a Petri net. PN is stable if there exists a Φ strictly positive m vector such that:*

$$\Delta v = u^T A\Phi \leq 0. \tag{6}$$

Moreover:

$$\Delta v = u^T A\Phi \leq 0 \Leftrightarrow A\Phi \leq 0.$$

Remark 31 *Notice that since the state space of a timed Petri net (TPN) is contained in the state space of the same now not timed PN, stability of PN implies stability of the TPN.*

Define as our vector Lyapunov function $v(M) = [v_1(M),\ v_2(M), ..., v_m(M)]^T$; where $v_i(M) = M(p_i), 1 \leq i \leq m$ then [6].

Proposition 32 *Let PN be a Petri net. PN is stabilizable if there exists a firing transition sequence with transition count vector u such that the following equation holds:*

$$\Delta v = A^T u \leq 0. \tag{7}$$

4 Discrete Event Time Dynamical Systems: A Case Study [6, 7]

Consider a two server queuing system (Fig 1.) whose timed Petri net model is depicted in Fig 2. Where the events (transitions) that drive the system are: q: customers arrive to the queue, s1, s2: service starts, d1,d2: the customer departs. The places (that represent the states of the queue) are: A: customers arriving, P: the customers are waiting for service in the queue, B1, B2: the customer is being served, I1, I2: the servers are idle. The holding times associated to the places A and I1, I2 are Ca and Cd respectively, (with $Ca > Cd$).

The incidence matrix that represents the PN model is:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

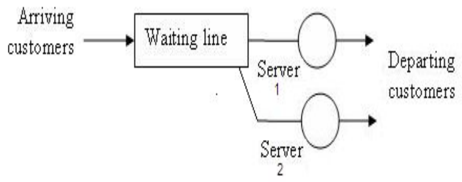

Fig. 1. Two server queuing system

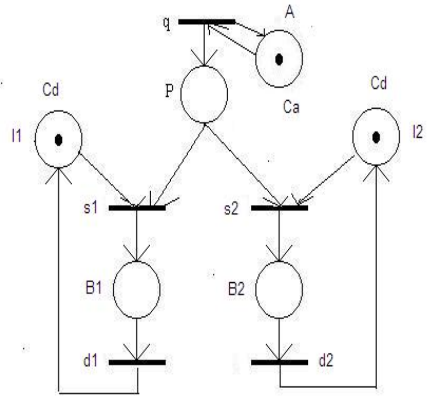

Fig. 2. Timed Petri net model

Therefore since there does not exists a Φ strictly positive m vector such that $A\Phi \leq 0$ the sufficient condition for stability is not satisfied.

Moreover, the PN (TPN) is unbounded since by the repeated firing of q, the marking in P grows indefinitely. However, by taking $u = [k, k/2, k/2, k/2, k/2]; k > 0$ we get that $A^T u \leq 0$. Therefore, the PN is stabilizable which implies that the TPN is stable i.e., the load has to be equally divided between the two servers. We have already discussed a $\psi - ASM$ whose program describes the discrete event time dynamical behavior of a Petri net (see 20, equation (3)), therefore setting in our program $m = 6, t = 5$, and $u = [k, k/2, k/2, k/2, k/2]$, we are able to bound the marking in P or equivalently to avoid the unstable states of the queuing system i.e., the set X_{un} of our discrete event time algorithm.

We conclude that by choosing properly the rules of the program $\psi - ASM$ the discrete event time algorithm for the two server queuing system is stabilizable.

Remark 33 *It has been shown by employing Max-Plus techniques, that we can set $k = Ca$ [7].*

5 Stability of Discrete Event Time Algorithms in Terms of Lyapunov Energy Functions for Discrete Event Time Algorithms

In this section, we consider the the states, (equivalently strucures), stability problem for discrete event time algorithms in terms of Lyapunov energy functions. The results presented in this section, become a generalization of what was discussed in section 3 and section 4, and includes them as particular cases. We will deal with discrete event time algorithms whose states are structures of vocabulary $\mathcal{V}$, where now the base set S is a metric space (S, d), with metric d.

Definition 34 *Let us consider a discrete event time algorithm, we will say that the state X, with $a \in S$ and time-indexed location $f_{t,t_0}(a)$, where t and t_0 belong to T, is stable if and only if $\forall t_0 \in T$ and $\forall \varepsilon > 0 \; \exists \; \delta = \delta(t_0, \varepsilon) > 0$ such that if given $a' \in S$, with $d(a', a) < \delta \Rightarrow d(\|f_{t,t_0}(a')\|_X, \|f_{t,t_0}(a)\|_X) < \varepsilon$ $\forall t \in T$.*

Definition 35 *Let us consider a discrete event time algorithm, we will say that the state X, with $a \in S$ and time-indexed location $f_t(a)$ is continuous at $t \in T$, if and only if $\forall \varepsilon > 0 \; \exists \; \delta = \delta(t) > 0$ and a state Y such that if given $t' \in T$, with $\mid t - t' \mid < \delta \Rightarrow d(\|f_t(a)\|_X, \|f_{t'}(a)\|_Y) < \varepsilon$.*

Definition 36 *A continuous function $\alpha : [0, \infty) \rightarrow [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$.*

Postulate E. The Lyapunov energy function associated to the Petri net model of the discrete event time algorithm at some transition firing time point $t_0 \in T$ multiplied by some finite constant $C \geq 1$ bounds the whole Lyapunov energy function, transferred or transformed of the whole discrete event time algorithm, as the Lyapunov energy function evolves in time.

Theorem 37 *Let us consider a discrete event time algorithm having transition firing time points $\{t_1, t_2, \cdots\} \subseteq T$. Assume there exists a Lyapunov function $V : S \times T \rightarrow \mathcal{R}^+$ and two functions α_1, α_2, which belong to $\mathcal{K}$, such that:*

$$\begin{aligned} \alpha_1(d(\|f_{t,t_0}(a')\|_X, \|f_{t,t_0}(a)\|_X)) &\leq \\ V(\|f_{t,t_0}(a')\|_X, t) &\leq \\ \alpha_2(d(\|f_{t,t_0}(a')\|_X, \|f_{t,t_0}(a)\|_X)) & \end{aligned}$$

for all $a, a' \in S$, $t, t_0 \in T$. Assume Postulate E and that $\|f_{t_0,t_0}(a')\|_X = a'$ holds, then the discrete event time algorithm is stable.

Proof. We want to show that there exists a $\delta = \delta(t_0, \varepsilon) > 0$ such that given a' with $d(a', a) < \delta \Rightarrow d(\|f_{t,t_0}(a')\|_X, \|f_{t,t_0}(a)\|_X) < \varepsilon$ $\forall t \in T$. Claim $\delta = \alpha_2^{-1}(C^{-1}\alpha_1(\varepsilon))$ does the job. $d(\|f_{t,t_0}(a')\|_X, \|f_{t,t_0}(a)\|_X) \leq \alpha_1^{-1}(V(\|f_{t,t_0}(a')\|_X, t)) \leq \alpha_1^{-1}(CV(\|f_{t_0,t_0}(a')\|_X, t_0)) = \alpha_1^{-1}(CV(a', t_0)) \leq \alpha_1^{-1}(C\alpha_2(d(a', a))) < \varepsilon$. Where postulate E has been used in the second inequality and the equation $\|f_{t_0,t_0}(a')\|_X = a'$ in the first equality. ■

6 Discrete Event Time Algorithm Reachability Tree Analysis

The lack of uniqueness of paths along the reachability tree of the Petri net which models a fixed discrete event time algorithm requires a little bit of extra analysis because, we need to take into account the possibility of multiple trajectories starting from each initial marking. This multiplicity leads us to consider the stability concept together with the adjectives *total* and *partial*. Roughly speaking, total is used when the stability property is satisfied for all trajectories starting from each initial marking. On the other hand, partial is used when the stability property is satisfied by at least one trajectory starting from each initial marking.

Definition 38 *A discrete event time algorithm for multiple paths along the reachability tree is the cartesian product of discrete event time algorithms, one for each path.*

Postulate F. A discrete event time algorithm for multiple paths along the reachability tree is a discrete event time dynamical system $\{T, X, A, \{\phi_{t_i}\}_{i\in\mathcal{N}}\}$, where T, X, A are defined as in Definition 1, $\{\phi_{t_i}\}_{i\in\mathcal{N}}$ is an indexed family of evolution operators parameterized by T, and for each $i \in \mathcal{N}$ fixed, there corresponds one member of the cartesian product of discrete event time algorithms.

Remark 39 *From definition (38) by taking cartesian products, it is immediate to generalize all the properties of discrete event time algorithms, (given in section 2), to discrete event time algorithms with multiple trajectories, where now we have an ASM program associated to each one of trajectories, and which defines the ASM program of the discrete event time algorithm for multiple paths along the reachability set.*

Next, we show how all the ideas presented above are applied in the following simple example.

Example 40 *Consider the discrete event time algorithm whose Petri net model is depicted in Fig 3.*

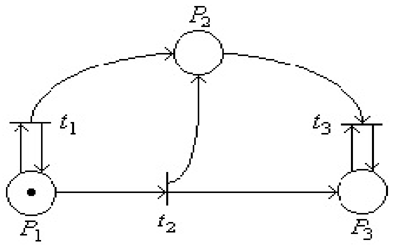

Fig. 3. Petri net model

The reachability tree of the Petri net, starting from the initial marking $(1, 0, 0)$ is shown in Fig 4.

This Petri net model has three paths along its reachability tree, one stable and two unstable. Therefore the discrete event time algorithm results to be partially stable. The ASM program of the discrete event time algorithm that models it, turns out to be composed by the following three ASM programs:

— $if \quad True \quad then \quad \phi_{n_1} := (1, n, 0),$

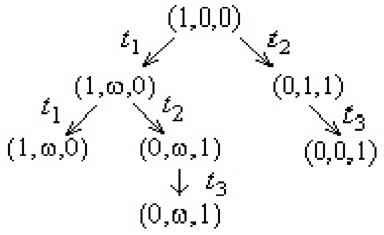

Fig. 4. Reachability tree

— $if \quad True \quad then \quad \phi_{n_2} := (0, n, 1),$

— $if \quad True \quad then \quad \phi_{n_3} := (0, 0, 1).$

The set of updates is given by $\triangle_\psi(X) = \{(\phi_{n_1} := (1, n, 0), (\phi_{n_2} := (0, n, 1)), (\phi_{n_3} := (0, 0, 1))\}.$

7 Conclusions

This paper introduces the axiomatization, computabilty and stability issues of discrete event time algorithms. Its main contribution consists in proposing a formalization free of any interpretation of discrete event time algorithms and continues discussing computability and stability issues. A Church type thesis, its proof and the notion of stabilty for discrete event time algorithms is proposed. The stability analysis presentation starts concentrating in discrete event time algorithms i.e., discrete event time dynamical systems, whose Petri net model is described by difference equations, and continues considering Lyapunov energy functions in terms of the logical structures of the vocabulary. A stability analysis based on the reachability tree of the Petri net model is also discussed.

References

1. **Gurevich, Y. (2000).** Sequential abstract-state machines capture sequential algorithms. *ACM Trans. Comput. Log.*, Vol. 1. DOI:10.1145/343369.343384.

2. **Dershowitz, N. & Gurevich, Y. (2008).** A natural axiomatization of computability and proof of Church's Thesis. *The Bulletin of Symbolic Logic*, Vol. 14, DOI:10.2178/bsl/1231081370.

3. **Bournez, O. & Dershowitz, N. (2016).** *Axiomatizing Analog Algorithms*.

4. **Retchkiman Königsberg, Z. & Dershowitz, N. (2019).** The Church thesis, its proof, and the notion of stability and stabilization for analog algorithms. *Communications in Applied Analysis*, Vol. 23, No. 2.

5. **Murata, T. (1989).** Petri nets: Properties, analysis, and applications. *Proc. IEEE*, Vol. 77. DOI:10.1109/5.24143.

6. **Retchkiman, Z. (2005).** Stability theory for a class of discrete event time dynamical systems modeled with Petri nets. *International Journal of Hybrid Systems*, Vol. 4, No. 1.

7. **Retchkiman, Z. (2012).** Timed Petri Net Modeling and Lyapunov/Max-Plus-Algebra Stability Analysis for a type of Queuing Systems by means of timed Petri nets, Lyapunov methods and max-plus algebra. *International Journal of Pure and Applied Mathematics*, Vol. 77 No. 3. DOI:10.12732/ijpam.v86i2.6.

8. **Lakshmikantham, V., Matrosov, V.M., & Sivasundaram, S. (1991).** Vector Lyapunov Functions and Stability Analysis of Nonlinear Systems. *Kluwer Academic Publ., Dordrecht*.

Article received on 12/09/2020; accepted on 01/10/2020.
Corresponding author is Zvi Retchkiman Königsberg.