Expressivity of Time-Varying Graphs

A. Casteigts[1], P. Flocchini[2], E. Godard[3], N. Santoro[4], and M. Yamashita[5].

[1] Université de Bordeaux, France – acasteig@labri.fr
[2] University of Ottawa, Canada – flocchin@eecs.uottawa.ca
[3] Université Aix-Marseille, France – emmanuel.godard@lif.univ-mrs.fr
[4] Carleton University, Ottawa, Canada – santoro@scs.carleton.ca
[5] Kyushu University, Fukuoka, Japan – mak@inf.kyushu-u.ac.jp §

Abstract. *Time-varying graphs* model in a natural way infrastructure-less highly dynamic systems, such as wireless ad-hoc mobile networks, robotic swarms, vehicular networks, etc. In these systems, a path from a node to another might still exist over time, rendering computing possible, even though at no time the path exists in its entirety. Some of these systems allow waiting (i.e., provide the nodes with store-carry-forward-like mechanisms such as local buffering) while others do not.
In this paper, we focus on the structure of the time-varying graphs modelling these highly dynamical environments. We examine the complexity of these graphs, with respect to waiting, in terms of their *expressivity*; that is in terms of the language generated by the feasible journeys (i.e., the "paths over time").
We prove that the set of languages $\mathcal{L}_{nowait}$ when no waiting is allowed contains all computable languages. On the other end, using algebraic properties of quasi-orders, we prove that $\mathcal{L}_{wait}$ is just the family of *regular* languages, even if the presence of edges is controlled by some arbitrary function of the time. In other words, we prove that, when waiting is allowed, the power of the accepting automaton drops drastically from being as powerful as a Turing machine, to becoming that of a Finite-State machine. This large gap provides a measure of the impact of waiting.
We also study *bounded waiting*; that is when waiting is allowed at a node for at most d time units. We prove that $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$; that is, the complexity of the accepting automaton decreases only if waiting is unbounded.

1 Introduction

1.1 Highly Dynamic Networks

In the past, the majority of the research on networks and distributed computing has been on *static* systems. The study of *dynamic* networks has focused extensively on systems where the dynamics are due to *faults* (e.g., node or edge deletions or additions); the faults however are limited in scope, bounded in number. Even in the field of self-stabilization, where the faults are extensive and possibly unbounded, the faults in the network structure are considered anomalies

§ This work has been partially supported by NSERC Discovery Grants, by Dr. Flocchini's University Research Chair, and by a PIMS grant.

with respect to the correct behaviour of the system. There are however systems where the instability never ends, the network is never connected, the changes are unbounded and occur continuously, where the changes are not anomalies but integral part of the nature of the system. Such *highly dynamic* systems are quite widespread, and becoming more ubiquitous. The most obvious class is that of wireless mobile ad hoc networks: the topology of the communication network, formed by having an edge between two entities when they are in communication range, changes continuously in time as the movement of the entities destroys old connections and creates new ones. These infrastructure-less highly dynamic networks, variously called *delay-tolerant*, *disruptive-tolerant*, *challenged*, *opportunistic*, have been long and extensively investigated by the engineering community and, more recently, by distributed computing researchers, especially with regards to the problems of broadcast and routing (*e.g.* [Zha06]). Similar highly dynamic conditions occur also when the mobility of the entities follows a predictable pattern, e.g. periodic or cyclic routes (*e.g.* [LW09]). Interestingly, similar complex dynamics occur also in environments where there is no mobility at all, e.g., in *social networks* (*e.g.* [KKW08]).

The highly dynamic features of these networks and their temporal nature is captured in a natural way by the model of *time-varying graphs* (TVG), or *evolving graphs*, where edges between nodes exist only at some times (*e.g.*, see [BFJ03,CCF09,CFQS12,Fer04]). A crucial aspect of dynamic networks, and obviously of time-varying graphs, is that a path from a node to another might still exist over time, even though at no time the path exists in its entirety. It is this fact that renders routing, broadcasting, and thus computing possible in spite of the otherwise unsurmountable difficulties imposed by the nature of those networks. Hence, the notion of "path over time", formally called *journey*, is a fundamental concept and plays a central role in the definition of almost all concepts related to connectivity in time-varying graphs. Examined extensively, under a variety of names (e.g., temporal path, schedule-conforming path, time-respecting path, trail), informally a journey is a walk $<e_1, e_2, ..., e_k>$ and a sequence of time instants $<t_1, t_2, ..., t_k>$ where edge e_i exists at time t_i and its latency ζ_i at that time is such that $t_{i+1} \geq t_i + \zeta_i$.

While the concept of journey captures the notion of "path over time" so crucial in dynamical systems, it does not yet capture additional limitations that some of these environments can impose on the use of the journeys. More specifically, there are systems that provide the entities with store-carry-forward-like mechanisms (e.g., local buffering); thus an entity wanting to communicate with a specific other entity at time t_0, can wait until the opportunity of communication presents itself. There are however environments where such a provision is not available (e.g., there are no buffering facilities), and thus waiting is not allowed. In TVG this distinction is the one between a *direct* journey where $\forall i, t_{i+1} = t_i + \zeta_i$, and an *indirect* journey where it is possible to have i such that $t_{i+1} > t_i + \zeta_i$.

In this paper, we focus on the structure of the time-varying graphs modelling these highly dynamical environments. We examine the complexity of these graphs, with respect to waiting, in terms of their *expressivity*, that is of the language defined by the journeys, and establish results showing the difference that the possibility of waiting creates.

1.2 Main Contributions

Given a dynamic network modeled as a time-varying graph $\mathcal{G}$, a journey in $\mathcal{G}$ can be viewed as a word on the alphabet of the edge labels; in this light, the class of feasible journeys defines the language $L_f(\mathcal{G})$ expressed by $\mathcal{G}$, where $f \in \{wait, nowait\}$ indicates whether or not indirect journeys are considered feasible by the environment. Note that in the highly dynamic networks context, we consider journeys where the transitions are guarded by possibly arbitrary (computable) functions of the time.

We focus on the sets of languages $\mathcal{L}_{nowait} = \{L_{nowait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ and $\mathcal{L}_{wait} = \{L_{wait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$, where $\mathcal{U}$ is the set of all time-varying graphs; that is, we look at the languages expressed when waiting is, or is not allowed. For each of these two sets, the complexity of recognizing any language in the set (that is, the computational power needed by the accepting automaton) defines the complexity of the environment.

We first study the expressivity of time-varying graphs when waiting is not allowed, that is the only feasible journeys are direct ones. We prove that the set $\mathcal{L}_{nowait}$ contains all computable languages. That is, we show that, for any computable language L, there exists a time-varying graph $\mathcal{G}$ with computable parameters such that $L = L_{nowait}(\mathcal{G})$.

We next examine the expressivity of time-varying graphs if indirect journey are allowed. We prove that $\mathcal{L}_{wait}$ is precisely the set of *regular* languages even if the presence and latence functions are arbitrary complex functions of the time. The proof is algebraic and based on order techniques, relying on a theorem by Harju and Ilie [HI98] that enables to characterize regularity from the closure of the sets from a well quasi-order. In other words, we prove that, when waiting is allowed, the power of the accepting automaton drops drastically from being as powerful as a Turing machine, to becoming that of a Finite-State Machine.

To better understand the impact of waiting on the expressivity of time-varying graphs, we then turn our attention to *bounded waiting*; that is when indirect journeys are considered feasible if the pause between consecutive edges in the journeys has a duration bounded by $d > 0$. In other words, at each step of the journey, waiting is allowed only for at most d time units. We examine the set $\mathcal{L}_{wait[d]}$ of the languages expressed by time-varying graphs when waiting is allowed up to d time units. We prove that for any fixed $d \geq 0$, $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$, which which implies that the expressivity of time-varying graphs is not affected by allowing waiting for a limited amount of time.

1.3 Related Work

The literature on dynamic networks and dynamic graphs could fill a volume. Here we briefly mention only some of the work most directly connected to the results of this paper.

The idea of representing dynamic graphs as a sequence of (static) graphs, called *evolving graph*, was introduced in [Fer04] to study basic dynamic network problems from a centralized point of view. The evolving graph views the dynamics of the system as a sequence of *global* snapshots (taken either in discrete steps or when events occur). The computationally equivalent model of *time-varying graph* (TVG), introduced in [CFQS12] and used here, views the dynamics of the system from the *local* point of view of the entities. Both viewpoints have been extensively employed in the analysis of basic problems such as routing, broadcasting, gossiping and other forms of information spreading (*e.g.* [AKL08,CFMS13]); to study problems of exploration in vehicular networks with periodic routes (*e.g.*, [FMS13,IW11]); to examine failure detectors and consensus (*e.g.*, [KLO10]); and in the investigations of emerging properties in social networks (*e.g.* [KKW08]). A characterization of classes of TVGs with respect to properties typically assumed in the research can be found in [CFQS12]. The impact of bounded waiting in dynamic networks has been investigated for exploration [IW11].

The closest concept to TVG-automata, defined in this paper, are the well-established *Timed Automata* proposed by [AD94] to model real-time systems. A timed automaton has real valued clocks and the transitions are guarded with finite comparisons on the clock values; with only one clock and no reset it is a TVG-automaton with 0 latency. Note that, even in the simple setting of timed automata, some key problems, like inclusion, are undecidable for timed languages in the non-deterministic case, while the deterministic case lacks some expressive power. Note that we focus here on the properties of the un-timed part of the journeys, and that, given the guards can be arbitrary functions, the reachability problem is obviously not decidable for TVG-automaton. We are here mainly interested in comparing expressivity of waiting and non-waiting in TVGs.

2 Definitions and Terminology

Time-varying graphs: A *time-varying graph* $\mathcal{G}$ is a quintuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where V is a finite set of entities or *nodes*; $E \subseteq V \times V \times \Sigma$ is a finite set of relations between these entities (*edges*), possibly labeled by symbols in an alphabet Σ. The system is studied over a given time span $\mathcal{T} \subseteq \mathbb{T}$ called *lifetime*, where $\mathbb{T}$ is the temporal domain (typically, $\mathbb{N}$ or $\mathbb{R}^+$ for discrete and continuous-time systems, respectively); $\rho : E \times \mathcal{T} \to \{0, 1\}$ is the *presence* function, which indicates whether a given edge is available at a given time; $\zeta : E \times \mathcal{T} \to \mathbb{T}$, is the *latency* function, which indicates the time it takes to cross a given edge if starting at a given date (the latency of an edge could vary in time). Both presence and latency are arbitrary computable functions. The directed edge-labeled graph $G = (V, E)$, called the *footprint* of $\mathcal{G}$, may contain loops, and it may have more than one edge between the same nodes, but all with different labels.

A path over time, or *journey*, is a sequence $\langle (e_1, t_1), (e_2, t_2), ..., (e_k, t_k) \rangle$ where $\langle e_1, e_2, \ldots, e_k \rangle$ is a walk in the footprint G, $\rho(e_i, t_i) = 1$ (for $1 \leq i < k$), and $\zeta(e_i, t_i)$ is such that $t_{i+1} \geq t_i + \zeta(e_i, t_i)$ (for $1 \leq i < k$). If $\forall i, t_{i+1} = t_i + \zeta(e_i, t_i)$

the journey is said to be *direct*, *indirect* otherwise. We denote by $\mathcal{J}^*(\mathcal{G})$ the set of all journeys in $\mathcal{G}$.

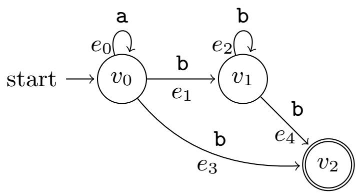
TVG-automata: Given a time-varying graph $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ whose edges are labeled over Σ, we define a TVG-automaton $\mathcal{A}(\mathcal{G})$ as the 5-tuple $\mathcal{A}(\mathcal{G}) = (\Sigma, S, I, \mathcal{E}, F)$ where Σ is the input *alphabet*; $S = V$ is the set of *states*; $I \subseteq S$ is the set of *initial states*; $F \subseteq S$ is the set of *accepting states*; $\mathcal{E} \subseteq S \times \mathcal{T} \times \Sigma \times S \times \mathcal{T}$ is the set of *transitions* such that $(s, t, a, s', t') \in \mathcal{E}$ iff $\exists e = (s, s', a) \in E : \rho(e, t) = 1, \zeta(e, t) = t' - t$. In the following we shall denote $(s, t, a, s', t') \in \mathcal{E}$ also by $s, t \overset{a}{\to} s', t'$. A TVG-automaton $\mathcal{A}(\mathcal{G})$ is *deterministic* if for any time $t \in \mathcal{T}$, any state $s \in S$, any symbol $a \in \Sigma$, there is at most one transition of the form $(s, t \overset{a}{\to} s', t')$; it is *non-deterministic* otherwise.

Given a TVG-automaton $\mathcal{A}(\mathcal{G})$, a *journey in* $\mathcal{A}(\mathcal{G})$ is a finite sequence of transitions $\mathcal{J} = (s_0, t_0 \overset{a_0}{\to} s_1, t_1), (s_1, t'_1 \overset{a_1}{\to} s_2, t_2) \ldots (s_{p-1}, t'_{p-1} \overset{a_{p-1}}{\to} s_p, t_p)$ such that the sequence $\langle(e_0, t_0), (e_1, t'_1), ..., (e_{p-1}, t'_{p-1})\rangle$ is a journey in $\mathcal{G}$ and $t_i = t'_{i-1} + \zeta(e_{i-1}, t'_{i-1})$, where $e_i = (s_i, s_{i+1}, a_i)$ (for $0 \leq i < p$). Consistently with the above definitions, we say that $\mathcal{J}$ is *direct* if $\forall i, t'_i = t_i$ (there is no pause between transitions), and *indirect* otherwise.We denote by $\lambda(\mathcal{J})$ the associated word $a_0, a_1, ...a_{p-1}$ and by $start(\mathcal{J})$ and $arrival(\mathcal{J})$ the dates t_0 and t_p, respectively. To complete the definition, an *empty* journey $\mathcal{J}_\emptyset$ consists of a single state, involves no transitions, its associated word is the empty word $\lambda(\mathcal{J}_\emptyset) = \varepsilon$, and its arrival date is the starting date. A journey is said *accepting* iff it starts in an initial state $s_0 \in I$ and ends in a accepting state $s_p \in F$. A TVG-automaton $\mathcal{A}(\mathcal{G})$ *accepts* a word $w \in \Sigma^*$ iff there exists an accepting journey $\mathcal{J}$ such that $\lambda(\mathcal{J}) = w$.

Let $L_{nowait}(\mathcal{G})$ denote the set of words (i.e., the *language*) accepted by TVG-automaton $\mathcal{A}(\mathcal{G})$ using only direct journeys, and let $L_{wait}(\mathcal{G})$ be the language recognized if journeys are allowed to be indirect. Given the set $\mathcal{U}$ of all possible TVGs, let us denote $\mathcal{L}_{nowait} = \{L_{nowait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ and $\mathcal{L}_{wait} = \{L_{wait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ the sets of all languages being possibly accepted by a TVG-automaton if journeys are constrained to be direct (i.e., no waiting is allowed) and if they are unconstrained (i.e., waiting is allowed), respectively.

In the following, when no ambiguity arises, we will use interchangeably the terms node and state, and the terms edge and transition; the term journey will be used both in reference to the sequence of edges in the TVG and to the corresponding sequence of transitions in the associated TVG-automaton.

Example of TVG-automaton: Figure 1a shows an example of a deterministic TVG-automaton that recognizes the context-free language $\{a^n b^n, n \geq 1\}$ (using only direct journeys). Consider the graph $\mathcal{G}_1 = (V, E, \mathcal{T}, \rho, \zeta)$, composed of three nodes: $V = \{v_0, v_1, v_2\}$, and five edges: $E = \{(v_0, v_0, a),\ (v_0, v_1, b),\ (v_1, v_1, b),\ (v_0, v_2, b),\ (v_1, v_2, b))\}$. The presence and latency functions are as shown in Table 1b, where p and q are two distinct prime numbers greater than 1. Consider now the corresponding automaton $\mathcal{A}(\mathcal{G}_1)$ where v_0 is the initial state and v_2 is the accepting state. For clarity, let us assume that $\mathcal{A}(\mathcal{G}_1)$ starts at time 1 (the same behavior could be obtained by modifying slightly the formulas involving t in Table 1b). It is clear that the a^n portion of the word $a^n b^n$ is read



(a) Structure of $\mathcal{G}_1$

e	$\rho(e,t) = 1$ iff	$\zeta(e,t) =$
e_0	always true	$(p-1)t$
e_1	$t > p$	$(q-1)t$
e_2	$t \neq p^i q^{i-1}, i > 1$	$(q-1)t$
e_3	$t = p$	any
e_4	$t = p^i q^{i-1}$, $i > 1$	any

(b) Presence and Latency functions for $\mathcal{G}_1$

Fig. 1: A TVG-automaton $\mathcal{G}_1$ such that $L_{nowait}(\mathcal{G}_1) = \{a^n b^n : n \geq 1\}$.

entirely at v_0 within $t = p^n$ time. If $n = 1$, at this time the only available edge is e_3 (labeled b) which allows to correctly accept ab. Otherwise ($n > 1$) at time $t = p^n$, the only available edge is e_1 which allows to start reading the b^n portion of the word. By construction of ρ and ζ, edge e_2 is always present except for the very last b, which has to be read at time $t = p^n q^{n-1}$. At that time, only e_4 is present and the word is correctly recognized. It is easy to verify that only these words are recognized, and the automaton is deterministic. The reader may have noticed the basic principle employed here (and later in the paper) of using latencies as a means to *encode* words into time, and presences as a means to *select* through opening the appropriate edges at the appropriate time.

3 No Waiting Allowed

This section focuses on the expressivity of time-varying graphs when only *direct* journeys are allowed. We prove that $\mathcal{L}_{nowait}$ includes all computable languages.

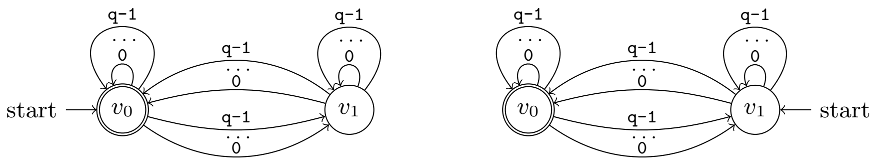
Let L be an arbitrary computable language defined over a finite alphabet Σ. Let ε denote the empty word; note that L might or might not contain ε. The notation $\alpha.\beta$ indicates the concatenation of $\alpha \in \Sigma^*$ with $\beta \in \Sigma^*$.

Let $q = |\Sigma|$ be the size of the alphabet, and w.l.o.g assume that $\Sigma = \{0, \ldots, q-1\}$. We define an injective encoding $\varphi : \Sigma^* \rightarrow \mathbb{N}$ associating to each word $w = a_0.a_1 \ldots a_k \in \Sigma^*$ the sum $q^{k+1} + \sum_{j=0}^{k} a_j q^{k-j}$. It is exactly the integer corresponding to $1.w$ interpreted in base q. By convention, $\varphi(\varepsilon) = 0$.

Consider now the TVG $\mathcal{G}_2$ where $V = \{v_0, v_1\}$, $E = \{\{(v_0, v_0, i), i \in \Sigma\} \cup \{(v_0, v_1, i), i \in \Sigma\} \cup \{(v_1, v_0, i), i \in \Sigma\} \cup \{(v_1, v_1, i), i \in \Sigma\}\}$. The presence and latency functions are defined relative to which node is the end-point of an edge. For all $u \in \{v_0, v_1\}$, $i \in \Sigma$, and $t \geq 0$, we define

- $\rho((u, v_0, i), t) = 1$ iff $t \in \varphi(\Sigma^*)$ and $\varphi^{-1}(t).i \in L$,
- $\zeta((u, v_0, i), t) = \varphi(\varphi^{-1}(t).i) - t$
- $\rho((u, v_1, i), t) = 1$ iff $t \in \varphi(\Sigma^*)$ and $\varphi^{-1}(t).i \notin L$,
- $\zeta((u, v_1, i), t) = \varphi(\varphi^{-1}(t).i) - t$

Consider the corresponding TVG-automaton $\mathcal{A}(\mathcal{G}_2)$ where the unique accepting state is v_0 and the initial state is either v_0 (if $\varepsilon \in L$, see Figure 2a), or v_1 (if $\varepsilon \notin L$ see Figure 2b).



(a) Case with $\varepsilon \in L$ (b) Case with $\varepsilon \notin L$

Fig. 2: A TVG $\mathcal{G}_2$ that recognizes an arbitrary computable language L.

Theorem 1. $L_{nowait}(\mathcal{G}_2) = L$.

Proof. First note that, since L is computable, testing for the appartenance to L in the definition of ρ and ζ is computable. Therefore the presence and latency function are computable.

Now, we want to show there is a unique accepting journey $\mathcal{J}$ with $\lambda(\mathcal{J}) = w$ iff $w \in L$. We first show that for all words $w \in \Sigma^*$, there is exactly one direct journey $\mathcal{J}$ in $\mathcal{A}(\mathcal{G}_2)$ such that $\lambda(\mathcal{J}) = w$, and in this case $arrival(\mathcal{J}) = \varphi(w)$. This is proven by induction on $k \in \mathbb{N}$, the length of the words. It clearly holds for $k = 0$ since the only word of that length is ε and $\varphi(\varepsilon) = 0$ (by convention, see above). Let $k \in \mathbb{N}$. Suppose now that for all $w \in \Sigma^*$, $|w| = k$ we have exactly one associated direct journey, and $arrival(\mathcal{J}) = \varphi(w)$.
Consider $w_1 \in \Sigma^*$ with $|w_1| = k + 1$. Without loss of generality, let $w_1 = w.i$ where $w \in \Sigma^*$ and $i \in \Sigma$. By induction there is exactly one direct journey $\mathcal{J}$ with $\lambda(\mathcal{J}) = w$. Let $u = arrival(\mathcal{J})$ be the node of arrival and t the arrival time. By induction, $t \in \varphi(\Sigma^*)$; furthermore since the presence function depends only on the node of arrival and not on the node of origin, there exists exactly one transition, labeled i from u. So there exists only one direct journey labeled by w_1. By definition of the latency function, its arrival time is $\varphi(\varphi^{-1}(t).i) = \varphi(w.i) = \varphi(w_1)$. This ends the induction.
We now show that such a unique journey is accepting iff $w \in L$. In fact, by construction of the presence function, every journey that corresponds to $w \in L, w \neq \varepsilon$, ends in v_0, which is an accepting state. By construction, the empty journey corresponding to ε ends in the accepting state v_0 if and only if $\varepsilon \in L$.

4 Waiting Allowed

We now turn the attention to the case of time-varying graphs where *indirect* journeys are possible. In striking contrast with the non-waiting case, we show that the languages $\mathcal{L}_{wait}$ recognized by TVG-automata consists only of regular languages. Let $\mathcal{R}$ denote the set of regular languages.

Lemma 1. $\mathcal{R} \subseteq \mathcal{L}_{wait}$.

Proof. It follows easily from observing that any finite-state machine (FSM) is a particular TVG-automaton whose edges are always present and have a nil latency. The fact that we allow waiting here does not modify the behavior of the automata as long as we consider deterministic FSMs only (which is sufficient), since at most one choice exists at each state for each symbol read. By considering exactly the same initial and final states, for any regular language L, we get a corresponding TVG $\mathcal{G}$ such that $L_{wait}(\mathcal{G}) = L$.

The reverse inclusion is more involved. Consider a non-deterministic automaton $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ with labels in Σ, we have to show that $L_{wait}(\mathcal{G}) \in \mathcal{R}$.

The proof is algebraic, and based on order techniques, relying on a theorem of Harju and Ilie (Theorem 6.3 in [HI98]) that enables to characterize regularity from the closure of the sets from a well quasi-order. We will use here an inclusion order on journeys (to be defined formally below). Informally, a journey $\mathcal{J}$ is included in another journey $\mathcal{J}'$ if its sequence of transitions is included (in the same order) in the sequence of transitions of $\mathcal{J}'$. It should be noted that sets of indirect journeys from one node to another are obviously closed under this inclusion order (on the journey $\mathcal{J}$ it is possible to wait on a node as if the missing transitions from $\mathcal{J}'$ were taking place), which is not the case for direct journeys as it is not possible to wait. In order to apply the theorem, we have to show that this inclusion order is a well quasi-order, i.e. that it is not possible to find an infinite set of journeys such that none of them could be included in another from the same set.

Let us first introduce some definitions and results about quasi-orders. We denote by $\leq$ a quasi-order over a given set Q. A set $X \subset Q$ is an *antichain* if all elements of X are pairwise incomparable. The quasi-order $\leq$ is *well founded* if in Q, there is no infinite descending sequence $x_1 \geq x_2 \geq x_3 \geq \ldots$ (where $\geq$ is the inverse of $\leq$) such that for no i, $x_i \leq x_{i+1}$. If $\leq$ is well founded and all antichains are finite then $\leq$ is a *well quasi-order* on Q. When $Q = \Sigma^*$ for alphabet Σ, a quasi-order is *monotone* if for all $x, y, w_1, w_2 \in \Sigma^*$, we have $x \leq y \Rightarrow w_1 x w_2 \leq w_1 y w_2$.

A word $x \in \Sigma^*$ is a *subword* of $y \in \Sigma^*$ if x can be obtained by deleting some letters on y. This defines a relation that is obviously transitive and we denote $\subseteq$ the *subword order* on Σ^*. Given two walks γ and γ', γ is a *subwalk* of γ', if γ can be obtained from γ' by deleting some edges. We can extend the $\subseteq$ order to labeled walks as follows: given two walks γ, γ' on the footprint G of $\mathcal{G}$, we note $\gamma \subseteq \gamma'$ if γ and γ' begin on the same node and end on the same node, and γ is a subwalk of γ'.

Given a date $t \in \mathcal{T}$ and a word x in Σ^*, we denote by $\mathcal{J}^*(t, x)$ the set $\{\mathcal{J} \in \mathcal{J}^*(\mathcal{G}) : start(\mathcal{J}) = t, \lambda(\mathcal{J}) = x\}$. $\mathcal{J}^*(x)$ denotes the set $\bigcup_{t \in \mathcal{T}} \mathcal{J}^*(t, x)$. Given a journey $\mathcal{J}$, $\bar{\mathcal{J}}$ is the corresponding labeled walk (in the footprint G). We denote by $\Gamma(x)$ the set $\{\bar{J} : \lambda(\mathcal{J}) = x\}$.

In the following, we consider only "complete" TVG so we have $\mathcal{J}^*(y)$ not empty for all word y; complete TVG can be obtained from any TVG (without changing the recognized language) by adding a sink node where any (missing) transition is sent. In this way, all words have at least one corresponding journey in the TVG.

Let x and y be two words in Σ^*. We define the quasi-order $\prec$, as follows: $x \prec y$ if

$$\forall \mathcal{J} \in \mathcal{J}^*(y), \exists \gamma \in \Gamma(x), \gamma \subseteq \bar{\mathcal{J}}.$$

The relation $\prec$ is obviously reflexive. We now establish the link between comparable words and their associated journeys and walks, and state some useful properties of relation $\prec$.

Lemma 2. *Let $x, y \in \Sigma^*$ be such that $x \prec y$. Then for any $\mathcal{J}_y \in \mathcal{J}^*(y)$, there exists $\mathcal{J}_x \in \mathcal{J}^*(x)$ such that $\bar{\mathcal{J}}_x \subseteq \bar{\mathcal{J}}_y$, $start(\mathcal{J}_x) = start(\mathcal{J}_y)$, $arrival(\mathcal{J}_x) = arrival(\mathcal{J}_y)$.*

Proof. By definition, there exists a labeled walk $\gamma \in \Gamma(x)$ such that $\gamma \subseteq \bar{\mathcal{J}}_y$. It is then possible to find a journey $\mathcal{J}_x \in \mathcal{J}^*(x)$ with $\bar{\mathcal{J}}_x = \gamma$, $start(\mathcal{J}_x) = start(\mathcal{J}_y)$ and $arrival(\mathcal{J}_x) = arrival(\mathcal{J}_y)$ by using for every edge of $\mathcal{J}_x$ the schedule of the same edge in $\mathcal{J}_y$.

Proposition 1. *The relation $\prec$ is transitive.*

Proof. Suppose we have $x \prec y$ and $y \prec z$. Consider $\mathcal{J} \in \mathcal{J}^*(z)$. By Lemma 2, we get a journey $\mathcal{J}_y \in \mathcal{J}^*(y)$, such that $\bar{\mathcal{J}}_y \subseteq \bar{\mathcal{J}}$. By definition, there exists $\gamma \in \Gamma(x)$ such that $\gamma \subseteq \bar{\mathcal{J}}_y$. Therefore $\gamma \subseteq \bar{\mathcal{J}}$, and finally $x \prec z$.

Let $L \subset \Sigma^*$. For any quasi-order $\leq$, we denote $\text{Down}_{\leq}(L) = \{x \mid \exists y \in L, x \leq y\}$.

The following is a corollary of Lemma 2:

Corollary 1. *Consider the language L of words induced by labels of journeys from u to v starting at time t. Then $\text{Down}_{\prec}(L) = L$.*

The following theorem is due to Harju and Ilie, this is a generalization of the well known theorem from Ehrenfeucht *et al* [EHR83], which needs closure in the other (upper) direction.

Theorem 2 (Th. 6.3 [HI98]). *For any monotone well quasi order $\leq$ of Σ^*, for any $L \subset \Sigma^*$, the language $\text{Down}_{\leq}(L)$ is regular.*

The main proposition to be proved now is that $(\Sigma^*, \prec)$ is a well quasi-order (Proposition 4 below). We have first to prove the following.

Proposition 2. *The quasi-order $\prec$ is monotone.*

Proof. Let x, y be such that $x \prec y$. Let $z \in \Sigma^*$. Let $\mathcal{J} \in \mathcal{J}^*(yz)$. Then there exists $\mathcal{J}_y \in \mathcal{J}^*(y)$ and $\mathcal{J}_z \in \mathcal{J}^*(arrival(\mathcal{J}_y), z)$ such that the end node of $\mathcal{J}_y$ is the start node of $\mathcal{J}_z$. By Lemma 2, there exists $\mathcal{J}_x$ that ends in the same node as $\mathcal{J}_y$ and with the same *arrival* time. We can consider $\mathcal{J}'$ the concatenation of $\mathcal{J}_x$ and $\mathcal{J}_z$. By construction $\bar{\mathcal{J}}' \in \Gamma(xz)$, and $\bar{\mathcal{J}}' \subseteq \bar{\mathcal{J}}$. Therefore $xz \prec yz$. The property $zx \prec zy$ is proved similarly using the *start* property of Lemma 2.

Proposition 3. *The quasi-order $\prec$ is well founded.*

Proof. Consider a descending chain $x_1 \succ x_2 \succ x_3 \succ \dots$ such that for no i $x_i \prec x_{i+1}$. We show that this chain is finite. Suppose the contrary. By definition of $\prec$, we can find $\gamma_1, \gamma_2, \dots$ such that for all i, $\gamma_i \in \bar{\mathcal{J}}^*(x_i)$, and such that $\gamma_{i+1} \subseteq \gamma_i$. This chain of walks is necessarily stationary and there exits i_0 such that $\gamma_{i_0} = \gamma_{i_0+1}$. Therefore, $x_{i_0} = x_{i_0+1}$, a contradiction.

To prove that $\prec$ is a well quasi-order, we now have to prove that all antichains are finite. Let $(Q, \leq)$ be a quasi-order. For all $A, B \subset Q$, we denote $A \leq_{\mathcal{P}} B$ if there exists an injective mapping $\varphi : A \longrightarrow B$, such that for all $a \in A$, $a \leq \varphi(a)$. The relation $\leq_{\mathcal{P}}$ is transitive and defines a quasi-order on $\mathcal{P}(Q)$, the set of subsets of Q.

About the finiteness of antichains, we recall the following result

Lemma 3 ([Hig52]). *Let $(Q, \leq)$ be a well quasi-order. Then $(\mathcal{P}(Q), \leq_{\mathcal{P}})$ is a well quasi-order.*

and the fundamental result of Higman:

Theorem 3 ([Hig52]). *Let Σ be a finite alphabet. Then $(\Sigma^*, \subseteq)$ is a well quasi-order.*

This implies that our set of journey-induced walks is also a well quasi-order for $\subseteq$ as it can be seen as a special instance of Higman's Theorem about the subword order. We are now ready to prove that all antichains are finite. We prove this result by using a technique similar to the variation by [Nas63] of the proof of [Hig52].

Lemma 4. *Let X be an antichain of Σ^*. If $\prec$ is a well quasi-order on $\text{Down}_\prec(X) \backslash X$ then X is finite or $\text{Down}_\prec(X) \backslash X = \emptyset$.*

Proof. We denote $Q = \text{Down}_\prec(X) \backslash X$, and suppose $Q \neq \emptyset$, and that Q is a well quasi-order for $\prec$. Therefore the product and the associated product order $(\Sigma \times Q, \prec_\times)$ define also a well quasi-order. We consider $A = \{(a, x) \mid a \in \Sigma, x \in Q, ax \in X\}$. Because $\prec$ is monotone, for all $(a, x), (a', x') \in A$, $(a, x) \prec_\times (b, y) \Rightarrow ax \prec by$. Indeed, in this case $a = b$ and $x \prec y \Rightarrow ax \prec ay$. So A has to be an antichain of the well quasi-order $\Sigma \times Q$. Therefore A is finite. By construction, this implies that X is also finite.

Theorem 4. *Let $L \subset \Sigma^*$ be an antichain for $\prec$. Then L is finite.*

Proof. Suppose we have an infinite antichain X_0. We apply recursively the previous lemma infinitely many times, that is there exists for all $i \in \mathbb{N}$, a set X_i that is also an infinite antichain of Σ^*, such that $X_{i+1} \subset \text{Down}_\prec(X_i) \backslash X_i$.

We remark that if we cannot apply the lemma infinitely many times that would mean that $X_k = \emptyset$ for some k. The length of words in X_0 would be bounded by k, hence in this case, finiteness of X_0 is also granted.

Finally, by definition of $\text{Down}_\prec$, for all $x \in X_{i+1}$, there exists $y \in X_i$ such that $x \prec y$, ie $x \subseteq y$. It is also possible to choose the elements x such that no pair is sharing a common y. So $X_{i+1} \subseteq_{\mathcal{P}} X_i$, and we have a infinite descending chain of $(\mathcal{P}(\Sigma^*), \subseteq_{\mathcal{P}})$. This would contradict Lemma 3.

From Propositions 1, 2, 3 and Theorem 4 we have the last missing ingredient:

Proposition 4. *$(\Sigma^*, \prec)$ is a well quasi-order.*

Indeed, from Proposition 4, Proposition 2, Corollary 1, and Theorem 2, it immediately follows that $L_{wait}(\mathcal{G})$ is a regular language for any TVG $\mathcal{G}$; that is,

Theorem 5. *$\mathcal{L}_{wait} = \mathcal{R}$.*

5 Bounded Waiting Allowed

To better understand the power of waiting, we now turn our attention to *bounded waiting*; that is when indirect journeys are considered feasible if the pause between consecutive edges has a bounded duration $d > 0$. We examine the set $\mathcal{L}_{wait[d]}$ of all languages expressed by time-varying graphs when waiting is allowed up to d time units, and prove the negative result that for any fixed $d \geq 0$, $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$. That is, the complexity of the environment is not affected by allowing waiting for a limited amount of time.

The basic idea is to reuse the same technique as in Section 3, but with a dilatation of time, i.e., given the bound d, the edge schedule is time-expanded by a factor d (and thus no new choice of transition is created compared to the no-waiting case).

Theorem 6. *For any duration d, $\mathcal{L}_{wait[d]} = \mathcal{L}_{wait[0]}$ (i.e., $\mathcal{L}_{nowait}$)*

Proof. Let L be an arbitrary computable language defined over a finite alphabet Σ. Let $d \in \mathbb{N}$ be the maximal waiting duration. We consider a TVG $\mathcal{G}_{2,d}$ structurally equivalent to $\mathcal{G}_2$ (see Figure 2 in Section 3), *i.e.,*, $\mathcal{G}_{2,d} = (V, E, \mathcal{T}, \rho, \zeta)$ such that $V = \{v_0, v_1, v_2\}$, $E = \{\{(v_0, v_1, i), i \in \Sigma\} \cup \{\{(v_0, v_2, i), i \in \Sigma\}, \cup \{(v_1, v_1, i), i \in \Sigma\} \cup \{(v_1, v_2, i), i \in \Sigma\} \cup \{(v_2, v_1, i), i \in \Sigma\} \cup \{(v_2, v_2, i), i \in \Sigma\}\}$. The initial state is v_0, and the accepting state is v_1. If $\varepsilon \in L$ then v_0 is also accepting.

Based on the mapping φ defined for $\mathcal{G}_2$ in Section 3, we define another mapping φ_d that associates to any word w the value $(d+1)\varphi(w)$. We also define $\psi_d(t)$ to be equal to $\varphi^{-1}(\lfloor \frac{t}{d+1} \rfloor)$ when it is defined. For instance, $\varphi_5(0110)$ in base 2 gives $(101 + 1) \times 10110$ (i.e., 132 in base 10). Reversely, we have $\psi_5(132) = ... = \psi_5(137) = 0110$, and $\psi_5(138) = ... = \psi_5(143) = 0111$.

The presence and latency functions are now defined along the lines as those of $\mathcal{G}_2$, the only difference being that we are using φ_d (resp. ψ_d) instead of φ (resp. φ^{-1}). Thus, for all $u \in \{v_0, v_1, v_2\}$, $i \in \Sigma$, and $t \geq 0$, we define

- $\rho((u, v_1, i), t) = 1$ iff $\lfloor \frac{t}{d+1} \rfloor \in \varphi_d(\Sigma^*)$ and $\psi_d(t).i \in L$,
- $\zeta((u, v_1, i), t) = \varphi_d(\psi_d(t).i) - t$
- $\rho((u, v_2, i), t) = 1$ iff $\lfloor \frac{t}{d+1} \rfloor \in \varphi_d(\Sigma^*)$ and $\psi_d(t).i \notin L$,
- $\zeta((u, v_2, i), t) = \varphi_d(\psi_d^{-1}(t).i) - t$

By the same induction technique as in Section 3, we have that $L \subseteq L(\mathcal{G}_{2,d})$. Similarly, we have that any journey labeled by w ends at time exactly $\varphi_d(w)$, even if some $d-$waiting occurred. Finally, we remark that for all words $w, w' \in \Sigma^+$ such that $w \neq w'$, we have $|\varphi_d(w) - \varphi_d(w')| > d$. Indeed, if $w \neq w'$ then they differ by at least one letter. The minimal time difference is when this is the last letter and these last letters are $i, i+1$ w.l.o.g. In this case, $|\varphi_d(w)-\varphi_d(w')| \geq d+1$ by definition of φ_d. Therefore waiting for a duration of d does not enable more transitions in terms of labeling.

References

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Comp. Sci.*, 126(2):183 – 235, 1994.

[AKL08] C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world. In *Proc. 35th Int. Coll, Automata Lang. Programming*, 121–132, 2008.

[BCF09] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proc. 28th Symp. Princ. Distr. Comput.*, 260–269, 2009.

[BFJ03] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Intl. J. Found. Comp. Science*, 14(2):267–285, 2003.

[CCF09] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proc. 16th Intl. Coll. Structural Information Comm. Complexity*, 126–140, 2009.

[CFMS13] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. *IEEE Transactions on Computers*, 2013.

[CFQS12] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[CMPS11] A. Clementi, A. Monti, F. Pasquale and R. Silvestri. Information spreading in stationary markovian evolving graphs. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1425–1432, 2011.

[EHR83] A. Ehrenfeucht, D. Haussler, and G. Rozenberg. On regularity of context-free languages. *Theoretical Computer Science*, 27(3):311–332, 1983.

[Fer04] A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.

[FMS13] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

[HI98] T. Harju and L. Ilie. On quasi orders of words and the confluence property. *Theoretical Computer Science*, 200(1-2):205 – 224, 1998.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2:326–336, 1952.

[IW11] D. Ilcinkas and A. Wade. On the power of waiting when exploring public transportation systems. In *15th Int. Conf. Princ. Distr. Sys.*, 451–464, 2011.

[KKW08] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network. In *Proc. 14th Intl. Conf. Knowledge Discovery Data Mining*, 435–443, 2008.

[KLO10] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proc. 42nd Symp. Theory Comp.*, 513–522, 2010.

[LW09] C. Liu and J. Wu. Scalable routing in cyclic mobile networks. *IEEE Trans. Parallel Distrib. Syst.*, 20(9):1325–1338, 2009.

[Nas63] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 59(04):833–835, 1963.

[Zha06] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.