Decidability of well-connectedness for distributed synthesis[☆]

Paul Gastin[a], Nathalie Sznajder[b]

[a] *LSV, ENS Cachan & CNRS & INRIA*
61, Av. du Président Wilson, F-94235 Cachan Cedex, France
[b] *LIP6, Université Pierre et Marie Curie & CNRS*
4 place Jussieu, 75005 Paris, France

Abstract

Although the synthesis problem is often undecidable for distributed, synchronous systems, it becomes decidable for the subclass of *uniformly well-connected (UWC)* architectures, provided that only robust specifications are considered. It is then an important issue to be able to decide whether a given architecture falls in this class. This is the problem addressed in this paper: we establish the decidability and precise complexity of checking this property. This problem is in EXPSPACE and NP-hard in the general case, but falls into PSPACE when restricted to a natural subclass of architectures.

1. Introduction

The synthesis problem, also known as Church's problem [3] consists in automatically deriving a system from its specification. The systems considered are open and reactive, i.e., maintain an ongoing interaction with an uncontrollable environment. For MSO specifications, the problem is decidable [2, 10]. The problem has then been studied for distributed systems [9]. In that case, along with the specification, is given a set of processes and a description of their possible interactions (the *architecture* of the system). Synthesis in the distributed case is a much more difficult problem, in general undecidable [9, 8] for LTL or CTL specifications. For some restricted subclasses of architectures though, the problem has been proved to be decidable [9, 6]. Following these results, a decidability criterion has been exhibited [4], when the specifications considered are in CTL* and *total*: they are allowed to constrain all the variables of the system, even those serving as internal communications between the processes. We have advocated in [5] that total specifications are too powerful, and that *external* specifications, that only relate input and output variables (those allowing communication with the environment) and let internal communications unconstrained, were both more natural from a practical point of view, and more likely to keep the problem decidable for a wider range of architectures. We have exhibited a decidability criterion for the subclass of uniformly well connected (UWC) architectures when restricted to external CTL* specifications, that indeed enlarge the classes of architectures for which the problem was known to be decidable. Moreover, we have shown that when restricted to *robust* external specifications, the synthesis problem is decidable for all UWC architectures. It is thus important to be able to decide whether a given architecture is uniformly well-connected. This is not trivial and has not been addressed in [5]. Informally, an architecture is UWC if there exists a routing allowing each output process to get, as soon as possible, the values of all inputs it is connected to. However, for the processes to decode the messages they receive and obtain the values of said inputs, we may need memory. If there is no bound on the size of this memory, a naive algorithm enumerating the different possible routings and decoding functions may not terminate. In this paper, we show that if an architecture is UWC, then there exist decoding functions with finite memories. We give an algorithm to decide whether an architecture is UWC. This can be done in EXPSPACE in the general case, and in PSPACE for some naturally restricted cases of the problem. We also make explicit the link between this notion and communication flow problems in network coding introduced in [1]. This relation al-

[☆] Partially supported by LIA INFORMEL.

lows us to establish that our problem is NP-hard.

2. Preliminaries

We use the formalism and notations defined in [5], simplified when possible to fit our purpose.

Architectures. An *architecture* $\mathcal{A} = (V \uplus P, E, (S^v)_{v\in V}, (d_p)_{p\in P})$ is a finite directed acyclic bipartite graph, where $V \uplus P$ is the set of vertices, and $E \subseteq (V \times P) \cup (P \times V)$ is the set of edges, such that $|E^{-1}(v)| \leq 1$ for all $v \in V$. Elements of P will be called *processes* and elements of V *variables*. Intuitively, an edge $(v, p) \in V \times P$ means that process p can read variable v, and an edge $(p, v) \in P \times V$ means that p can write on v. Thus, $|E^{-1}(v)| \leq 1$ means that a variable v is written by at most one process. Input and output variables are defined, respectively, by $V_I = \{v \in V \mid E^{-1}(v) = \emptyset\}$, $V_O = \{v \in V \mid E(v) = \emptyset\}$. Variables in $V_L = V \setminus (V_I \cup V_O)$ will be called *local* or *internal*. We assume that no process is minimal or maximal in the graph.

Each variable v ranges over a finite domain S^v, given with the architecture. When $U \subseteq V$, S^U will denote $\prod_{v\in U} S^v$. A *configuration* of the architecture is given by a tuple $s = (s^v)_{v\in V} \in S^V$ describing the value of all variables. For $U \subseteq V$, we denote by $s^U = (s^v)_{v\in U}$ the projection of the configuration s to the variables in U. The above notation is extended to words in the natural way: for $\sigma = s_1 \cdots s_n \in (S^V)^*$, its projection on $U \subseteq V$ is $\sigma^U = s_1^U \cdots s_n^U$. Also, we denote by $\sigma[i]$ the prefix of length i of σ (by convention, $\sigma[i] = \varepsilon$ if $i \leq 0$).

We will assume that $0 \in S^v$ and $|S^v| \geq 2$ for all $v \in V$. In fact, if $S^v = \{0\}$ then variable v would always have the same value, and could be ignored.

Each process $p \in P$ is associated with a delay $d_p \in \mathbb{N}$ that corresponds to the time interval between the moment the process reads the variables $v \in E^{-1}(p)$ and the moment it will be able to write on its output variables in $E(p)$. Note that delay 0 is allowed. For $v \in V \setminus V_I$ with $E^{-1}(v) = \{p\}$, we let $R(v) = E^{-1}(p) = E^{-2}(v)$ be the set of variables v directly depends on and $d_v = d_p$ be its delay.

An example of an architecture is given in Figure 1, where variables are represented by circles and processes by boxes with delays written next to them.

Semantics. We consider a discrete time, synchronous semantics. Informally, at each step the environment provides new values for input variables. Then, each process p reading values of variables in $E^{-1}(p)$ written at time $i - d_p$, computes values for variables in $E(p)$ and writes them at time i.

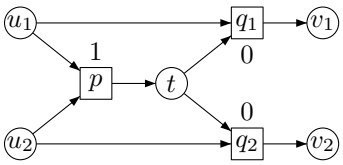

Figure 1: An architecture

Program, routing. The values of a local variable $v \in V_L$ will be computed by a *memoryless* program (or strategy) $f^v : S^{R(v)} \to S^v$. A *routing* for architecture $\mathcal{A}$ is a family $\Phi = (f^v)_{v\in V_L}$ of *memoryless* strategies. A Φ-sequence is a word $\sigma = s_1 \cdots s_n \in (S^{V\setminus V_O})^*$ such that for all $v \in V_L$ we have $s_i^v = 0$ for all $1 \leq i \leq d_v$ and $s_i^v = f^v(s_{i-d_v}^{R(v)})$ for all $d_v < i \leq |\sigma| = n$. This ensures that the delay d_v is respected when computing the next value of variable v. When the delays are all equal to 0, we also say that a configuration s is Φ-compatible when $s^v = f^v(s^{R(v)})$ for all $v \in V_L$.

View. For a variable $v \in V$, we let $\mathrm{InView}(v) = (E^{-2})^*(v) \cap V_I$ be the set of *input* variables v might depend on.

Delays. The *smallest cumulative delay* of transmission from u to v is inductively defined by $d(u, v) = +\infty$ if $v \notin (E^2)^*(u)$, i.e., if there is no path from u to v in the architecture, $d(u, u) = 0$, and $d(u, v) = d_v + \min\{d(u, w) \mid w \in (E^2)^+(u) \cap E^{-2}(v)\}$ for $u \neq v \in (E^2)^+(u)$.

We then say that an architecture is uniformly well-connected if there exists a routing Φ that allows to transmit with a minimal delay to every process p writing to an output variable v, the values of the variables in $\mathrm{InView}(v)$.

Definition 1. An architecture $\mathcal{A}$ is *uniformly well-connected* (UWC) if there exist a routing Φ and, for every $v \in V_O$ and $u \in \mathrm{InView}(v)$, a decoding function $g^{u,v} : \left(S^{R(v)}\right)^+ \to S^u$ that can reconstruct the value of u as soon as possible, i.e., such that for any Φ-sequence $\sigma = s_1 \cdots s_n \in (S^{V\setminus V_O})^+$ with $|\sigma| = n > d(u, v) - d_v$, we have

$$s^u_{n-d(u,v)+d_v} = g^{u,v}(\sigma^{R(v)}) \qquad (1)$$

In case there is no delay, the uniform well-connectedness refines the notion of adequate connectivity introduced by Pnueli and Rosner in [9],

as we no longer require each output variable to be communicated the value of *all* input variables, but only those in its view.

As already pointed out in [5], the definition of uniform well-connectedness is not symmetric: whereas the routing functions are memoryless, some memory is required for the decoding functions. We take the example given in [5] to prove this fact. Consider the architecture from Figure 1 and assume that all variables range over the domain $\{0,1\}$. It is clear that this architecture is UWC: process p writes to t the xor of u_1 and u_2 with delay 1. This could be written $t = \mathsf{Y}\, u_1 \oplus \mathsf{Y}\, u_2$ where $\mathsf{Y}\, x$ denotes the previous value of variable x. In order to recover (decode) $\mathsf{Y}\, u_2$, process q_1 stores the previous value of u_1 and makes the xor with t: $\mathsf{Y}\, u_2 = t \oplus \mathsf{Y}\, u_1$. But if we restrict to memoryless decoding functions, then we only know u_1 and t and we cannot recover $\mathsf{Y}\, u_2$.

Let us show formally that if we restrict to memoryless decoding functions, we cannot recover the values of the input variables. Fix a routing $\Phi = f^t$, where f^t is memoryless, and fix decoding functions $(g^{u,v})_{v\in V_O, u\in \mathrm{InView}(v)}$ satisfying (1). Note that, if $s_1 s_2 \in (S^V)^2$ is a Φ-sequence, then s_2^t only depends on $s_1^{\{u_1,u_2\}}$ since f^t is memoryless and $d_p = 1$. Now, f^t cannot be injective, hence we find another Φ-sequence $s_1' s_2' \in (S^V)^2$ with $s_2^t = s_2'^t$ and $s_1^{\{u_1,u_2\}} \neq s_1'^{\{u_1,u_2\}}$ and $s_2^{\{u_1,u_2\}} = s_2'^{\{u_1,u_2\}}$. For instance $s_1^{u_2} \neq s_1'^{u_2}$ and we get $g^{u_2,v_1}\big((s_1 s_2)^{\{u_1,t\}}\big) = s_1^{u_2} \neq s_1'^{u_2} = g^{u_2,v_1}\big((s_1' s_2')^{\{u_1,t\}}\big)$. It follows that g^{u_2,v_1} is not memoryless.

However, and interestingly, while proving the decidability of this criterion, we will establish that if an architecture is uniformly well-connected, then decoding functions with *finite* memory suffice. By that, we mean a function of the form $g : M \times S^{R(v)} \to S^v$, where M is a finite memory that can be updated at each step. Such functions will be represented by finite automata with outputs.

3. Decidability and complexity of checking UWC

Observe that, since the routing functions are memoryless, the value of any variable in a Φ-sequence is influenced only by a limited number of input values in the past. This is expressed by the following lemma.

First, we introduce some additional notations. Here we use not only the minimal transmission delay $d(u,v)$ from u to v but also the maximal transmission delay $D(u,v)$ defined by $D(u,v) = +\infty$ if $v \notin (E^2)^*(u)$, $D(u,u) = 0$, and $D(u,v) = d_v + \max\{D(u,w) \mid w \in (E^2)^+(u) \cap E^{-2}(v)\}$ for $u \neq v \in (E^2)^+(u)$. If $\sigma = s_1 s_2 \cdots s_n$ is a sequence and i, j are integers then $\sigma[i \cdot\cdot j]$ denotes the subsequence $s_i \cdots s_j$, which is empty if $i > j$. We also use this notation when $i \leq 0$ or $j \leq 0$: if $j \leq 0$ then $\sigma[i \cdot\cdot j] = \varepsilon$ is the empty sequence, and if $i \leq 0 < j$ then $\sigma[i \cdot\cdot j] = \sigma[1 \cdot\cdot j]$.

Lemma 2. *For all $v \in V \setminus V_O$, for any routing Φ and any Φ-sequence $\sigma = s_1 \dots s_i$, the value s_i^v only depends on $(\sigma^u[i - D(u,v) \cdot\cdot i - d(u,v)])_{u\in InView(v)}$.*

Proof. The proof is by induction on the variables of the acyclic architecture. It is trivial for input variables and also when $i \leq d_v$ since in this case we have $s_i^v = 0$. Suppose now $v \in V_L$ and $i > d_v$. Since σ is a Φ-sequence, we have $s_i^v = f^v(s_{i-d_v}^{R(v)})$. Let $w \in R(v)$. By induction hypothesis, $s_{i-d_v}^w$ depends only on $(\sigma^u[i - d_v - D(u,w) \cdot\cdot i - d_v - d(u,w)])_{u\in \mathrm{InView}(w)}$. We have $\mathrm{InView}(v) = \bigcup_{w\in R(v)} \mathrm{InView}(w)$. Hence, using the definitions of the minimal and maximal transmission delays, we deduce that s_i^v depends only on $(\sigma^u[i - D(u,v) \cdot\cdot i - d(u,v)])_{u\in \mathrm{InView}(v)}$. □

Corollary 3. *For all $v \in V_O$, for all routing Φ, and all Φ-sequence $\sigma = s_1 \cdots s_i$, the value $s_i^{R(v)}$ only depends on $(\sigma^u[i + d_v - D(u,v) \cdot\cdot i + d_v - d(u,v)])_{u\in InView(v)}$.*

Proof. Let $w \in R(v)$. By Lemma 2, s_i^w only depends on $(\sigma^u[i - D(u,w) \cdot\cdot i - d(u,w)])_{u\in \mathrm{InView}(w)}$. For $u \in \mathrm{InView}(w)$, we have $D(u,w) \leq D(u,v) - d_v$ and $d(u,w) \geq d(u,v) - d_v$. The result follows since $\mathrm{InView}(w) \subseteq \mathrm{InView}(v)$. □

According to Corollary 3, given a routing Φ, for each output variable $v \in V_O$, we can define a global coding function Ψ_v which computes the values of variables in $R(v)$ from a limited input segment.

Formally, fix $v \in V_O$. For any input sequence $\rho = r_1 \cdots r_i \in (S^{V_I})^*$, and $u \in \mathrm{InView}(v)$, we define $m_v^u(\rho) = \rho^u[i + d_v - D(u,v) \cdot\cdot i - 1 + d_v - d(u,v)]$ and $\mathrm{lst}_v^u(\rho) = r_{i+d_v-d(u,v)}^u$ if $i > d(u,v) - d_v$ and $\mathrm{lst}_v^u(\rho) = \varepsilon$ otherwise. Moreover, we let $m_v(\rho) = (m_v^u(\rho))_{u\in \mathrm{InView}(v)}$ and $\mathrm{lst}_v(\rho) = (\mathrm{lst}_v^u(\rho))_{u\in \mathrm{InView}(v)}$.

Note that, if $\sigma = s_1 \cdots s_i$ is the Φ-sequence induced by ρ, i.e., such that $\sigma^{V_I} = \rho$, then the

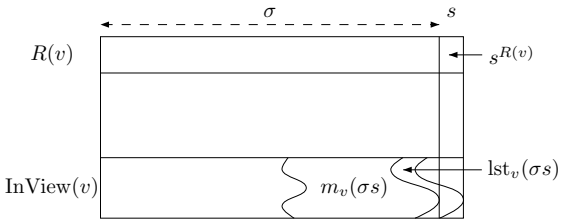

Figure 2: Fragment of the input influencing the value $s^{R(v)}$

value $s_i^{R(v)}$ only depends on $(m_v(\rho), \mathrm{lst}_v(\rho))$ by Corollary 3. To simplify the notation, we let $m_v(\sigma) = m_v(\rho)$ and $\mathrm{lst}_v(\sigma) = \mathrm{lst}_v(\rho)$. Figure 2 illustrates this notion: the bottom rectangle represents $(\sigma s)^{\mathrm{InView}(v)}$. The curved shape picturing $m_v(\sigma s)$ and $\mathrm{lst}_v(\sigma s)$ shows the input values on which $s^{R(v)}$ depends (note that $\sigma^{R(v)}$ does *not* depend on $\mathrm{lst}_v(\sigma s)$). The shape is irregular because of the different minimal and maximal transmission delays between variables in $\mathrm{InView}(v)$ and variables in $R(v)$.

Let the domain $\mathrm{dom}(\Psi_v)$ of the global coding function be the set of pairs $(m_v(\rho), \mathrm{lst}_v(\rho))$ where ρ is an input sequence. Note that $\mathrm{dom}(\Psi_v)$ is finite. Moreover, if $\sigma = s_1 \cdots s_i$ is the Φ-sequence induced by an input sequence ρ, we let $\Psi_v(m_v(\rho), \mathrm{lst}_v(\rho)) = s_i^{R(v)}$. Note that, if $\sigma' = s'_1 \cdots s'_j$ is any Φ-sequence such that $(m_v(\sigma'), \mathrm{lst}_v(\sigma')) = (m_v(\sigma), \mathrm{lst}_v(\sigma))$ then $s'^{R(v)}_j = s_i^{R(v)}$ by Corollary 3. Therefore, Ψ_v is well-defined.

The following lemma gives a necessary and sufficient condition for a routing to be decodable:

Lemma 4. *Given a routing Φ, there exist decoding functions $(g^{u,v})_{v \in V_O, u \in InView(v)}$ satisfying Eq. (1) of Definition 1 if and only if, for all $v \in V_O$, for all $(m, y), (m, y') \in \mathrm{dom}(\Psi_v)$, we have*

$$\Psi_v(m, y) = \Psi_v(m, y') \implies y = y' \qquad (2)$$

Informally, the coding function Ψ_v computes, from some history m and a new input value y, the value $s^{R(v)} = \Psi_v(m, y)$ that will reach the decoding functions $(g^{u,v})_{u \in \mathrm{InView}(v)}$. If (2) holds then it is possible to recover the new input value y from the finite memory m and the value $s^{R(v)}$. Hence, decoding functions with finite memory exist. Conversely, if there are some (m, y), (m, y') such that $\Psi_v(m, y) = \Psi_v(m, y')$, we will build two Φ-sequences σ and σ' that coincide on the values in $R(v)$. If decoding functions exist (not necessarily with finite memories) they will compute the same input value $y = y'$ from σ and σ'. The formal proof follows.

Proof of Lemma 4. Assume that there exist decoding functions $(g^{u,v})_{v \in V_O, u \in \mathrm{InView}(v)}$ satisfying Eq. (1) of Definition 1. Let $(m, y), (m, y') \in \mathrm{dom}(\Psi_v)$ be such that $\Psi_v(m, y) = \Psi_v(m, y')$.

Let $\sigma = s_1 \cdots s_i$ be a Φ-sequence such that $(m, y) = (m_v(\sigma), \mathrm{lst}_v(\sigma))$. Then, $\Psi_v(m, y) = s_i^{R(v)}$.

Let $\rho = r_1 \cdots r_i \in (S^{V_I})^i$ be an input sequence such that for all $u \in \mathrm{InView}(v)$ we have $\rho^u[1 \cdots i-1+d_v-d(u,v)] = \sigma^u[1 \cdots i-1+d_v-d(u,v)]$ and $r^u_{i+d_v-d(u,v)} = y'^u$ if $i > d(u,v) - d_v$. Let $\sigma' = s'_1 \cdots s'_i \in (S^{V \setminus V_O})^+$ be the (unique) Φ-sequence induced by ρ, i.e., such that $\sigma'^{V_I} = \rho$. By definition, we have $(m, y') = (m_v(\sigma'), \mathrm{lst}_v(\sigma'))$. Hence, $\Psi_v(m, y') = s'^{R(v)}_i$.

Applying Corollary 3, we get $\sigma^{R(v)}[1 \cdots i-1] = \sigma'^{R(v)}[1 \cdots i-1]$ since for all $u \in \mathrm{InView}(v)$ we have $\sigma'^u[1 \cdots i - 1 + d_v - d(u,v)] = \sigma^u[1 \cdots i - 1 + d_v - d(u,v)]$. Moreover, $s_i^{R(v)} = \Psi_v(m, y) = \Psi_v(m, y') = s'^{R(v)}_i$. Hence $\sigma^{R(v)} = \sigma'^{R(v)}$.

We deduce that for all $u \in \mathrm{InView}(v)$ such that $i > d(u,v) - d_v$, we have $y^u = s^u_{i+d_v-d(u,v)} = g^{u,v}(\sigma^{R(v)}) = g^{u,v}(\sigma'^{R(v)}) = s'_{i+d_v-d(u,v)} = y'^u$. Therefore, $y = y'$.

Now, fix some $v \in V_O$ and assume that (2) holds for all $(m, y), (m, y') \in \mathrm{dom}(\Psi_v)$. We will define the decoding functions $(g^{u,v})_{u \in \mathrm{InView}(v)}$ using a finite automaton $\mathfrak{A}_v$ whose set of states Q_v defines the finite memory of the decoding functions.

First, for $u \in \mathrm{InView}(v)$ and σ a Φ-sequence, we define $M^u_v(\sigma) = \sigma^u[|\sigma|+1+d_v-D(u,v) \cdots |\sigma|+d_v-d(u,v)]$ and $M_v(\sigma) = (M^u_v(\sigma))_{u \in \mathrm{InView}(v)}$. Note the shift by 1 between m_v and M_v: if σs is a Φ-sequence, then $M_v(\sigma) = m_v(\sigma s)$ and $M_v(\sigma s)$ is determined by $M_v(\sigma)$ and $\mathrm{lst}_v(\sigma s)$ (see Figure 2). The decoding functions will recover the value $\mathrm{lst}_v(\sigma s)$ from $s^{R(v)}$ that it can directly read, and from its memory $M_v(\sigma) = m_v(\sigma s)$.

Then, Q_v is the set of all $M_v(\sigma)$ such that σ is a Φ-sequence. The initial state is $M_v(\varepsilon) = (\varepsilon)_{u \in \mathrm{InView}(v)}$. The set of transitions consists of all $M_v(\sigma) \xrightarrow{s^{R(v)}} M_v(\sigma s)$ such that σs is a Φ-sequence. Clearly, for each Φ-sequence σ, there is a run of $\mathfrak{A}_v$ starting from the initial state, reading $\sigma^{R(v)}$, and leading to state $M_v(\sigma)$. This run is unique since $\mathfrak{A}_v$ is deterministic as we show next.

Let σs and $\sigma' s'$ be two Φ-sequences such that $M_v(\sigma) = m = M_v(\sigma')$ and $s^{R(v)} = s'^{R(v)}$. Let $y = \mathrm{lst}_v(\sigma s)$ and $y' = \mathrm{lst}_v(\sigma' s')$. We have $\Psi_v(m, y) = s^{R(v)} = s'^{R(v)} = \Psi_v(m, y')$ since $m_v(\sigma s) = m =$

$m_v(\sigma' s')$. Hence we get $y = y'$ from our hypothesis. We deduce that $M_v(\sigma s) = M_v(\sigma' s')$ since this value is determined by $(M_v(\sigma), \mathrm{lst}_v(\sigma s)) = (M_v(\sigma'), \mathrm{lst}_v(\sigma' s'))$. Therefore, $\mathfrak{A}_v$ is deterministic.

Finally, for a Φ-sequence σs, we define $(g^{u,v}((\sigma s)^{R(v)}))_{u \in \mathrm{InView}(v)}$ as the unique y such that $\Psi_v(m_v(\sigma s), y) = s^{R(v)}$. Recall that $\mathfrak{A}_v$ allows to compute $m_v(\sigma s) = M_v(\sigma)$ as the state reached when reading $\sigma^{R(v)}$ from the initial state. Moreover, $\Psi_v(m_v(\sigma s), \mathrm{lst}_v(\sigma s)) = s^{R(v)}$. Hence $y = \mathrm{lst}_v(\sigma s)$ and (1) is satisfied. □

From the above results, we deduce:

Theorem 5. *The problem of checking whether a given architecture is UWC is decidable. It is in* EXPSPACE *in the general case, and in* PSPACE *if we restrict to architectures which are 0-delay, where the size of the variable domains is fixed, and where the read-degree is fixed.*

Proof. Let $\mathcal{A} = (V \uplus P, E, (S^v)_{v \in V}, (d_p)_{p \in P})$ be an architecture. For each $v \in V_O$ we define $\Delta_v = 1 + \max\{D(u, v) - d_v \mid u \in \mathrm{InView}(v)\}$. First we claim that to get $\mathrm{dom}(\Psi_v)$ it suffices to consider input sequences of length at most Δ_v. Indeed, let $\rho = r_1 \cdots r_i \in (S^{V_I})^*$ be an input sequence of length $i > \Delta_v$ and let $\rho' = r_{i+1-\Delta_v} \cdots r_i$. We can easily check that $m_v(\rho') = m_v(\rho)$ and $\mathrm{lst}_v(\rho') = \mathrm{lst}_v(\rho)$ which proves our claim.

We check whether $\mathcal{A}$ is UWC with the following non-deterministic procedure. Guess a routing $\Phi = (f^v)_{v \in V_L}$. For each $v \in V_O$ and each pair $\rho, \rho' \in (S^{V_I})^+$ of input sequences of length at most Δ_v such that $m_v(\rho') = m_v(\rho)$, compute the Φ-sequences $\sigma = s_1 \cdots s_i$ induced by ρ and $\sigma' = s'_1 \cdots s'_j$ induced by ρ' and check that if $s_i^{R(v)} = s_j'^{R(v)}$ then $\mathrm{lst}_v(\rho') = \mathrm{lst}_v(\rho)$. The correctness of the algorithm follows immediately from Lemma 4.

We establish now the complexity upper bound. Considering that we write in binary the cardinal of each domain S^v and the value of each delay d_p, the space needed to store the architecture is $\mathrm{sz}(\mathcal{A}) = A + B + C$ where

$$A = \sum_{v \in V} \lceil \log_2(1 + |S^v|) \rceil \qquad B = \sum_{p \in P} \lceil \log_2(1 + d_p) \rceil$$

and $C = 2|V||P|$ is the space needed to store E. We first compute the space $\mathrm{sz}(\Phi)$ needed to store a routing Φ. For each variable $v \in V_L$, the space needed to store the memoryless routing $f^v : S^{R(v)} \to S^v$ is $|S^{R(v)}| \cdot \lceil \log_2(1 + |S^v|) \rceil$. Hence,

$$\mathrm{sz}(\Phi) = \sum_{v \in V_L} |S^{R(v)}| \cdot \lceil \log_2(1 + |S^v|) \rceil \ .$$

Next, fix $v \in V_O$. The space needed to store a Φ-sequence σ of length at most Δ_v is bounded by $A\Delta_v$ since the space needed for a configuration $s \in S^V$ is bounded by A. The same holds for the input sequences ρ, ρ' of length at most Δ_v used in our decision procedure. Then our non-deterministic procedure uses space bounded by $\mathrm{sz}(\Phi) + 4 \cdot A \cdot \max_{v \in V_O}(\Delta_v)$.

More precisely, in the general case, $|S^{R(v)}| \leq |S^V| \leq 2^A$ hence Φ can be stored within exponential space. Moreover, $\Delta_v \leq 1 + \sum_{p \in P} d_p \leq \prod_{p \in P}(1 + d_p) \leq 2^B$. Hence, σ can be stored with exponential space. We deduce that the non-deterministic decision procedure uses exponential space in the general case. The result follows since NEXPSPACE = EXPSPACE.

We assume now that $d_p = 0$ for all $p \in P$, and that for all $v \in V$ we have $|S^v| \leq c_s$ and $|R(v)| \leq c_r$ where c_s and c_r are constants which do not depend on the input. The space needed to store Φ becomes

$$\mathrm{sz}(\Phi) \leq |V| \cdot c_s^{c_r} \cdot \lceil \log_2(1 + c_s) \rceil = \mathcal{O}(\mathrm{sz}(\mathcal{A}))$$

since $|V| \leq |E| \leq \mathrm{sz}(\mathcal{A})$. Therefore, in that case, the routing Φ can be stored within polynomial space.

Since all delays are 0, we have $\Delta_v = 1$ for all $v \in V_O$. Hence we only consider input sequences of length 1 and the problem is in NPSPACE = PSPACE.

More precisely, in this restricted case, the problem is in fact in $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$. Recall that, in the polynomial hierarchy, the complexity class $\mathrm{NP}^{\mathrm{NP}}$ corresponds to problems that can be solved by a non-deterministic Turing machine working in polynomial time, with the help of an oracle for a problem in NP. Given a routing Φ, we call a pair $r, r' \in S^{V_I}$ *conflicting* if for the corresponding Φ-compatible configurations $s, s' \in S^{V \setminus V_O}$ and for some $v \in V_O$, we have $s^{R(v)} = s'^{R(v)}$ but $r^{\mathrm{InView}(v)} \neq r'^{\mathrm{InView}(v)}$. Such a pair is a witness of *incorrectness* of Φ. Given an architecture $\mathcal{A}$ and a routing Φ, the problem of deciding the existence of a conflicting pair is in NP: first guess the two input values r and r', guess the output variable v, compute the corresponding Φ-configurations s and

s' and check the property. Note that a configuration $s \in S^{V \setminus V_O}$ can be stored in space $\log_2(|S^V|) \leq |V| \cdot \log_2(c_s) = \mathcal{O}(\mathrm{sz}(\mathcal{A}))$. We can check that, in polynomial time, we can guess the pair r, r', compute the corresponding Φ-configurations s, s', and checking the property above.

Now, our procedure works as follows: guess a routing Φ and ask the oracle whether there is a conflicting pair for Φ or not. The routing can be guessed in polynomial time (and stored in polynomial space). Hence the problem is in $\mathrm{NP}^{\mathrm{NP}}$. □

4. Relation to Network Coding

We now show the interesting relationship between uniform well-connectedness and the network information flow problem, introduced in [1]. Instances of such problems are directed acyclic graphs in which two subsets of nodes have been distinguished: the *sources* and the *sinks*. Along with such a graph comes a certain amount of *messages*, and each sink demands a subset of the messages. Formally, an instance of the network information flow problem is a tuple $(P, E, M, S, \mathrm{demand})$ where M is the set of messages (input variables), P is the set of processes, $E \subseteq (P \times P) \cup (M \times P)$ is the edge relation defining an acyclic graph and the set $V = E \cap (P \times P)$ corresponds to the (implicit) internal variables of the network. All variables in $M \cup V$ of the network have the same domain S. A process is a source if it is connected to some input message, i.e., the set of sources is $E(M)$. Finally, the map $\mathrm{demand} : P \to 2^M$ defines what messages should be routed to which processes. A sink is a process $p \in P$ with $\mathrm{demand}(p) \neq \emptyset$. We impose that $\mathrm{demand}(p) \subseteq (E^{-1})^*(p) \cap M = \mathrm{InView}(p)$. For $p \in P$, let $R(p) = (M \cap E^{-1}(p)) \cup (E \cap (P \times \{p\}))$, moreover, let $R(v) = R(p)$ for $v = (p, q) \in V$. The solution of such a problem is a *0-delay routing* $\Phi = (f^v)_{v \in V}$ with $f^v : S^{R(v)} \to S$ and decoding functions $g^{m,p} : S^{R(p)} \to S$ for $p \in P$ and $m \in \mathrm{demand}(p)$ such that, for any Φ-compatible configuration s, we have $g^{m,p}(s^{R(p)}) = s^m$. We say then that Φ satisfies the demands.

One specific problem that has been more extensively studied in that area is the *multicast*: an instance of the aforementioned problem in which there is a unique source, and every sink demands all messages.

The networks considered in information flow problems mainly differ from our architectures on the following aspects. First, a variable is attached to an edge, hence there is exactly one process that can read each variable whereas in our case several processes might read the same variable. Second, there is a single (uniform) domain for all variables of the network instance of an information flow problem whereas we may have domains with different sizes for the variables of our architectures. And last, there is no delay in the transmission of information while we may add various delays to our processes. Hence, our architectures are more flexible and we get:

Lemma 6. *There is a polynomial reduction from the multicast problem to the uniform well-connectedness problem.*

Proof. Let $\mathcal{A} = (P, E, M, S, \mathrm{demand})$ be an instance of the multicast problem. We have $E(M) = \{p_0\}$ where p_0 is the unique source. We show how to build a corresponding architecture that would be UWC if and only if there is a 0-delay routing solving the original multicast problem. The messages will be turned into values of input variables, and explicit variables will be added: one input variable for each message of $\mathcal{A}$, one output variable for each sink process of $\mathcal{A}$, and one internal variable for each edge of $\mathcal{A}$.

Formally, we define an architecture $\mathcal{A}' = (V' \uplus P, E', (S^v)_{v \in V'}, (d_p)_{p \in P})$ by $V' = V'_I \uplus V'_O \uplus V'_L$ with $V'_I = M$, $V'_O = \{v_p \mid p \text{ is a sink}\}$ and $V'_L = V$, $E' = (M \times \{p_0\}) \cup \{(p, v_p) \mid p \text{ is a sink}\} \cup \bigcup_{v=(p,q) \in V} \{(p, v), (v, q)\}$, $S^v = S$ for all $v \in V'$, and $d_p = 0$ for all $p \in P$. Observe that for all $v \in V'_O$, we have $\mathrm{InView}(v) = V'_I = M$ and for all $v \in V'_L$ we have $R'(v) = R(v)$. Hence, the notion of 0-delay routing on $\mathcal{A}$ coincides with the notion of routing on $\mathcal{A}'$.

Next, we have seen in the proof of Lemma 4 that if decoding functions exist for $\mathcal{A}'$ then they only need finite memory computed by the automata $\mathfrak{A}_v$. But when all delays are 0 we have $|Q_v| = 1$, which means that the decoding functions $g^{u,v}$ are memoryless. In this case, Condition (1) of Definition 1 coincides with the condition on the memoryless decoding functions for $\mathcal{A}$.

Hence, the multicast problem for $\mathcal{A}$ coincides with the UWC problem for $\mathcal{A}'$. □

This reduction allows us to deduce a complexity lower bound for checking uniform well-connectedness. The multicast problem was shown to be NP-hard in [11]. Since it can be reduced to our problem by Lemma 6, we obtain:

Corollary 7. *The problem of checking whether a given architecture is UWC is NP-hard.*

To be complete on the relationship between information flow and uniform well-connectedness, we also show that, for some restricted architectures, the reduction is also valid in the other direction.

Lemma 8. *The uniform well-connectedness problem for architectures that are 0-delay, and in which all the variables range over the same domain S can be reduced to the network information flow problem.*

Proof. Since in a network instance of the information flow problem, only one process can read a variable whereas a same variable can be read by several processes in our architectures, we will replace variables of an architecture by processes of the network.

Let $\mathcal{A} = (V \uplus P, E, S)$ be such an architecture. Consider $\mathcal{A}' = (P', E', M', S, \text{demand})$ be an instance of the network information flow problem where $P' = P \cup V_L$, $E' = E \setminus (P \times V_O)$, $M' = V_I$ and for all $p \in E^{-1}(V_O)$, $\text{demand}(p) = \text{InView}(v)$ where $v \in E(p)$ is arbitrary (they all have the same input-view). The set of internal variables V' is then $V' = E' \cap (P' \times P') = E \setminus (P \times V_O \cup V_I \times P) = P \times V_L \cup V_L \times P$.

Suppose there is a routing $\Phi = (f^v)_{v \in V_L}$ on $\mathcal{A}$ and, for every $v \in V_O$, a decoding function $g^v : S^{R(v)} \to S^{\text{InView}(v)}$. We have to simulate the routing $\Phi = (f^v)_{v \in V_L}$ of the architecture $\mathcal{A}$ with a new routing $\Phi' = (f'^e)_{e \in V'}$ on the network $\mathcal{A}'$. If $e = (p, v) \in P \times V_L$ then, on $\mathcal{A}$, process p writes on variable v, which is simulated on $\mathcal{A}'$ by setting $f'^e = f^v$. In the other case, the previous value is simply copied to the next edge. Formally, let $\Phi' = (f'^e)_{e \in V'}$ be the routing on $\mathcal{A}'$ defined as follows : for $r \in S^{R'(e)}$

- if $e = (p, v) \in P \times V_L$, then $R'(e) = (V_I \cap R(v)) \cup \{(u, p) \mid u \in V_L \cap R(v)\}$ so we have $S^{R'(e)} = S^{R(v)}$ and we can set $f'^e(r) = f^v(r)$,
- if $e = (v, p) \in V_L \times P$, then $R'(e)$ is a singleton $\{(q, v)\}$ where q is the unique process writing in v in $\mathcal{A}$ and we set $f'^e(r) = r$.

For all $p \in E^{-1}(V_O)$ sink of the network, we let $g'^p = g^v$, for some $v \in V_O \cap E(p)$, and show that they are decoding functions for the network $\mathcal{A}'$.

Let $s' \in S^{M' \cup V'}$ be a Φ'-configuration and $s \in S^{V \setminus V_O}$ defined by $s^{V_I} = s'^{M'}$ and, for all $v \in V_L$, $s^v = s'^{(p,v)}$, where $p \in P$ is the unique process writing on v.

We show that s is a Φ-configuration. Let $v \in V_L$, then by definition, $s^v = s'^{(p,v)} = f'^{(p,v)}(s'^{R'(p,v)}) = f^v(s'^{R'(p,v)})$. We have seen that $R'(p, v) = (V_I \cap R(v)) \cup \{(u, p) \mid u \in V_L \cap R(v)\}$. For $e = u \in V_I \cap R(v)$, we have $s'^e = s^u$ by definition of s. Let $e = (u, p)$ with $u \in V_L \cap R(v)$. We have seen that $R'(e) = \{(q, u)\}$, where q is the unique process writing on u in $\mathcal{A}$. Then, $s'^e = f'^e(s'^{R'(e)}) = s'^{(q,u)} = s^u$ by definition of s. Then, $s'^{R'(p,v)} = s^{R(v)}$ and $s^v = f^v(s^{R(v)})$.

Let $p \in E^{-1}(V_O)$ and $v \in V_O \cap E(p)$. As above we can show that $s'^{R'(p)} = s^{R(v)}$. We deduce that $g'^p(s'^{R'(p)}) = g^v(s^{R(v)}) = s^{\text{InView}}(v)$.

Conversely, let $\Phi' = (f'^e)_{e \in V'}$ be a routing for $\mathcal{A}'$. We define the routing $\Phi = (f^v)_{v \in V_L}$ as follows. Let $v \in V_L$ be some internal variable and let $p \in E^{-1}(v)$ be the unique process which writes on v. We have seen that $R'(p, v) = (V_I \cap R(v)) \cup \{(u, p) \mid u \in V_L \cap R(v)\}$. Hence, $S^{R(v)} = S^{R'(p,v)}$ and we can set $f^v(r) = f'^{(p,v)}(r)$ for all $r \in S^{R(v)}$.

Assume there are decoding functions $(g'^p)_{p \in E^{-1}(V_O)}$ that satisfy the demand. Let $v \in V_O$ and $p \in E^{-1}(v)$. We can see as above that $S^{R(v)} = S^{R'(p)}$ and we can define $g^v(r) = g'^p(r)$ for $r \in S^{R(v)}$. We show that $(g^v)_{v \in V_O}$ yield uniform well-connectedness.

Let s be a Φ-outcome and s' be the Φ'-outcome induced by the same input values: $s'^{V_I} = s^{V_I}$. We show by induction that for all $v \in V_L$, we have $s^v = s'^{(p,v)}$ where $p \in E^{-1}(v)$. Let $v \in V_L$ and $p \in E^{-1}(v)$. Since $R'(p, v) = (V_I \cap R(v)) \cup \{(u, p) \mid u \in V_L \cap R(v)\}$, we obtain $s^{R(v)} = s'^{R'(p,v)}$ by induction and using $s'^{V_I} = s^{V_I}$. Then, we get $s^v = f^v(s^{R(v)}) = f'^{(p,v)}(s^{R(v)}) = f'^{(p,v)}(s'^{R'(p,v)}) = s'^{(p,v)}$.

Finally, for each $v \in V_O$ and $p \in E^{-1}(p)$, we get as above $s^{R(v)} = s'^{R'(p)}$. Then $g^v(s^{R(v)}) = g'^p(s'^{R'(p)}) = s'^{\text{InView}(v)} = s^{\text{InView}(v)}$ showing that the decoding functions are correct. □

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *Trans. Inf. Theory*, 46(4):1204–1216, 2000.

[2] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:295–311, 1969.

[3] A. Church. Logic, arithmetics, and automata. In *Proc. of Int. Congr. of Mathematicians*, pages 23–35, 1962.

[4] B. Finkbeiner and S. Schewe. Uniform distributed synthesis. In *Proc. of LICS'05*, pages 321–330. IEEE Computer Society Press, 2005.

[5] P. Gastin, N. Sznajder, and M. Zeitoun. Distributed synthesis for well-connected architectures. *Formal Meth. Syst. Des.*, 34(3):215–237, 2009.

[6] O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *Proc. of LICS'01*. IEEE Computer Society Press, 2001.

[7] P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *Proc. of ICALP'01*, volume 2076 of *LNCS*, pages 396–407. Springer, 2001.

[8] G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proc. of FOCS'79*, pages 348–363. IEEE Computer Society Press, 1979.

[9] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of FOCS'90*, volume II, pages 746–757. IEEE Computer Society Press, 1990.

[10] M. O. Rabin. *Automata on Infinite Objects and Church's Problem*. Amer. Math. Soc., Boston, MA, USA, 1972.

[11] A. Rasala Lehman and E. Lehman. Complexity classification of network information flow problems. In *Proc. of SODA'04*, pages 142–150. SIAM, 2004.