Infinite Networks, Halting and Local Algorithms

Antti Kuusisto*
Institute of Computer Science
University of Wrocław
Poland
antti.j.kuusisto@uta.fi

The immediate past has witnessed an increased amount of interest in local algorithms, i.e., constant time distributed algorithms. In a recent survey of the topic (Suomela, ACM Computing Surveys, 2013), it is argued that local algorithms provide a natural framework that could be used in order to theoretically control infinite networks in finite time. We study a comprehensive collection of distributed computing models and prove that if infinite networks are included in the class of structures investigated, then every universally halting distributed algorithm is in fact a local algorithm. To contrast this result, we show that if only finite networks are allowed, then even very weak distributed computing models can define nonlocal algorithms that halt everywhere. The investigations in this article continue the studies in the intersection of logic and distributed computing initiated in (Hella et al., PODC 2012) and (Kuusisto, CSL 2013).

1 Introduction

This work is a study of deterministic distributed algorithms for arbitrary networks, including infinite structures in addition to finite ones. In the recent survey article [13], Suomela points out that distributed constant-time algorithms are a reasonable choice for theoretically controlling infinite networks in finite time. In this article we show that for a rather comprehensive collection of models of distributed computing, constant-time algorithms are in a sense the *only* choice. We define a framework—based on a class of message passing automata and relational structures—that contains a comprehensive variety of models of distributed computing in *anonymous networks*, i.e., networks without ID-numbers. We then show that if infinite networks are allowed, then *all universally halting algorithms* definable in the framework are in fact local algorithms, i.e., distributed constant-time algorithms.

The widely studied *port-numbering model* (see [2, 8, 9]) of distributed computing can be directly extended to a framework that contains infinite structures in addition to finite ones. In the port-numbering model, a node of degree $k \leq n$, where n is a globally known finite degree bound, receives messages through k input ports and sends messages through k output ports. The processors in the nodes can send different messages to different neighbours, and also see from which port incoming messages arrive. There are no ID-numbers in this framework. The omission of ID-numbers is well justified when infinite networks are studied: in most natural theoretical frameworks for the modelling of computation in infinite networks, *even the reading of all local* IDs *in the beginning of computation would take infinitely long*. Thus typical synchronized communication using ID-numbers would be impossible.

There are several fields of study outside distributed computing where the objects of investigation can be regarded as infinite distributed anonymous communication networks. *Cellular automata* provide

*The author acknowledges that this work was carried out during a tenure of the ERCIM "Alain Bensoussan" Fellowship Programme. The reported research has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 246016.

Adriano Peron and Carla Piazza (Eds.):
Proceedings of the Fifth International Symposium on
Games, Automata, Logics and Formal Verification (GandALF 2014)
EPTCS 161, 2014, pp. 147–160, doi:10.4204/EPTCS.161.14

probably the most obvious and significant example of such a framework. But of course there are various others. Crystal lattices and the brain, for example, are massive network systems often modelled by infinite structures.

Below we define a *general distributed computing model* based on relational structures and synchronized message passing automata. The port-numbering model $\mathrm{VV_c}$ of [8, 9] and all its subclasses can be directly simulated in our framework by restricting attention to suitable classes of structures and automata. We establish (Theorem 4.3) that if $\mathscr{H}$ is a class of communication networks definable by a *first-order theory*, then all universally halting algorithms over $\mathscr{H}$ are local algorithms. For example, the classes of networks for the VV_c model are easily seen to be definable by first-order formulae, *as long as infinite structures are allowed*. In fact, when the requirement of finiteness is lifted, all classes of structures in the comprehensive study in [8, 9] can easily be seen to be first-order definable.

The proof of Theorem 4.3 makes a *crucial use of logic*, thereby extending the work initiated in [8, 9] and developed further in [11]. The articles [8, 9, 11] extend the scope of *descriptive complexity theory* (see [7, 10, 12]) to the realm of distributed computing by identifying a highly canonical one-to-one link between *local* algorithms and formulae of modal logic. This link is based on the novel idea of directly identifying *Kripke models* and distributed communication networks with each other. Under this interpretation, arrows of the accessibility relations of Kripke models are considered to ***be*** communication channels between processors in distributed networks. This idea has turned out to be fruitful because it enables the transfer of results between modal logic and distributed computing. For example, in [8, 9] a novel separation argument concerning distributed complexity classes is obtained by applying the *bisimulation method* (see [4, 5, 6]) of modal logic to distributed communication networks.

In this article we adapt the link between modal logic and distributed computing for the purpose of proving Theorem 4.3. We first obtain a characterization of halting behaviour in terms of modal formulae. This facilitates the use of the *compactness theorem* (see [7]), which is the final step in our proof.

To contrast Theorem 4.3, we investigate halting behaviour of distributed message passing automata in the finite. We establish that even extremely weak subsystems of the port-numbering model can define nonlocal halting algorithms when attention is restricted to finite networks: Theorem 3.1 shows that even if message passing automata in the port-numbering model have absolutely no access to port numbers whatsoever, nonlocal but universally halting behaviour is possible.

In order to prove Theorem 3.1, we employ tools from *combinatorics on words*, namely, the infinite *Thue-Morse sequence* (see [1]). This infinite binary sequence is known to be cube-free, i.e., it does not have a prefix of the type $tuuu$, where u is a nonempty word. This lack of periodicity allows us to design an appropriate algorithm that is halting but nonlocal in the finite.

2 Preliminaries

Let Π be a finite set of *unary relation symbols* $P \in \Pi$ and $\mathscr{R}$ a finite set of *binary relation symbols* $R \in \mathscr{R}$. These symbols are also called *predicate symbols*. The set of $(\Pi,\mathscr{R})$*-formulae* of *modal logic* $\mathrm{ML}(\Pi,\mathscr{R})$ is generated by the grammar

$$\varphi \ ::= \ \top \mid P \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \langle R \rangle \varphi,$$

where P is any symbol in Π, R any symbol in $\mathscr{R}$, and $\top$ is a logical constant symbol. Let $\mathrm{VAR} = \{ x_i \mid i \in \mathbb{N} \}$ be a set of *variable symbols*. The set of $(\Pi,\mathscr{R})$*-formulae* of *first-order logic* $\mathrm{FO}(\Pi,\mathscr{R})$ is generated by the grammar

$$\varphi \ ::= \ \top \mid x = y \mid P(x) \mid R(x,y) \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \exists x \varphi,$$

where x and y are symbols in VAR, P a symbol in Π, R a symbol in $\mathscr{R}$, and $\top$ a logical constant symbol. For both logics, we define the abbreviation $\bot := \neg\top$. We also use the abbreviation symbols $\vee$, $\rightarrow$ and $\leftrightarrow$ in the usual way. The *modal depth* $md(\varphi)$ of a formula is defined recursively such that $md(\top) = md(P) = 0$, $md(\neg\psi) = md(\psi)$, $md(\psi \wedge \chi) = max\{md(\psi), md(\chi)\}$, and $md(\langle R\rangle\psi) = md(\psi)+1$.

Let $\Pi = \{P_1, ..., P_n\}$ and $\mathscr{R} = \{R_1, ..., R_m\}$. A $(\Pi, \mathscr{R})$-*model* is a structure

$$M := (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M),$$

where W is an arbitrary nonempty set (the *domain* of the model M), each P_i^M is a unary relation $P_i^M \subseteq W$, and each R_i^M a binary relation $R_i^M \subseteq W \times W$. The semantics of $\mathrm{ML}(\Pi, \mathscr{R})$ is defined with respect to *pointed* $(\Pi, \mathscr{R})$-*models* (M, w), where $M = (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M)$ is a $(\Pi, \mathscr{R})$-model and $w \in W$ a *point* or a *node* of (the domain of) M. For each $P_i \in \Pi$, we define $(M, w) \models P_i$ iff $w \in P_i^M$. We also define $(M, w) \models \top$. We then recursively define

$$\begin{array}{lcl} (M,w) \models \neg\varphi & \Leftrightarrow & (M,w) \not\models \varphi, \\ (M,w) \models (\varphi \wedge \psi) & \Leftrightarrow & (M,w) \models \varphi \text{ and } (M,w) \models \psi, \\ (M,w) \models \langle R_i\rangle\varphi & \Leftrightarrow & \exists v \in W\big((w,v) \in R_i^M \text{ and } (M,v) \models \varphi\big). \end{array}$$

The semantics of $\mathrm{FO}(\Pi, \mathscr{R})$ is defined in the usual way with respect to $(\Pi, \mathscr{R})$-*interpretations* (M, f), where

$$M = (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M)$$

is a $(\Pi, \mathscr{R})$-model and f is an *assignment function* $f : \mathrm{VAR} \to W$ giving an interpretation to the variables in VAR. We define $(M, f) \models x = y \Leftrightarrow f(x) = f(y)$, $(M, f) \models P_i(x) \Leftrightarrow f(x) \in P_i^M$, and $(M, f) \models R_i(x, y) \Leftrightarrow (f(x), f(y)) \in R_i^M$. We also define $(M, f) \models \top$. We then recursively define

$$\begin{array}{lcl} (M,f) \models \neg\varphi & \Leftrightarrow & (M,f) \not\models \varphi, \\ (M,f) \models (\varphi \wedge \psi) & \Leftrightarrow & (M,f) \models \varphi \text{ and } (M,f) \models \psi, \\ (M,f) \models \exists x\varphi & \Leftrightarrow & \exists v \in W\big((M, f[x \mapsto v]) \models \varphi\big), \end{array}$$

where $f[x \mapsto v]$ is the function $g : \mathrm{VAR} \to W$ such that

$$g(y) = \begin{cases} v & \text{if } y = x, \\ f(y) & \text{if } y \neq x. \end{cases}$$

It is well known that modal logic can be directly translated into first-order logic. We define the *standard translation* from $\mathrm{ML}(\Pi, \mathscr{R})$ into $\mathrm{FO}(\Pi, \mathscr{R})$ in the following way. We let $St_x(\top) := \top$, $St_x(P_i) := P_i(x)$, $St_x((\varphi \wedge \psi)) := \big(St_x(\varphi) \wedge St_x(\psi)\big)$, $St_x(\neg\varphi) := \neg St_x(\varphi)$, and $St_x(\langle R_i\rangle\varphi) := \exists y\big(R_i(x,y) \wedge St_y(\varphi)\big)$. Here y is a fresh variable distinct from x. It is easy to see that $(M, w) \models \varphi$ iff $(M, f[x \mapsto w]) \models St_x(\varphi)$. Due to the standard translation, modal logic is often considered to be simply a *fragment* of first-order logic.

We next fix some conventions concerning *sets* of formulae. We only discuss formulae of first-order logic, but analogous definitions hold for modal logic.

If Φ is a set of formulae of $\mathrm{FO}(\Pi, \mathscr{R})$, then $\bigvee\Phi$ and $\bigwedge\Phi$ denote the *disjunction* and *conjunction* of the formulae in Φ. The set Φ can be infinite, but then of course neither $\bigvee\Phi$ nor $\bigwedge\Phi$ is a formula of $\mathrm{FO}(\Pi, \mathscr{R})$. We define $(M, f) \models \bigvee\Phi$ if there exists at least one formula $\varphi \in \Phi$ such that $(M, f) \models \varphi$. We define $(M, f) \models \bigwedge\Phi$ if $(M, f) \models \varphi$ for all $\varphi \in \Phi$. A *set* of formulae of $\mathrm{FO}(\Pi, \mathscr{R})$ is called a *theory* (over the signature $(\Pi, \mathscr{R})$).[1] If T is a theory over the signature $(\Pi, \mathscr{R})$, then $(M, f) \models T$ means that $(M, f) \models \varphi$

[1] A theory does not have to be closed under logical consequence. A theory is simply a set of formulae, and can be infinite or finite.

for all $\varphi \in T$. When we write $T \models \varphi$, we mean that the implication $(M,f) \models T \Rightarrow (M,f) \models \varphi$ holds for all $(\Pi,\mathscr{R})$-interpretations (M,f). As usual, two FO$(\Pi,\mathscr{R})$-formulae φ and ψ are *equivalent* if the equivalence $(M,f) \models \varphi \Leftrightarrow (M,f) \models \psi$ holds for all $(\Pi,\mathscr{R})$-interpretations (M,f).

Let $\mathscr{H}$ be a class of *pointed* $(\Pi,\mathscr{R})$-models, and let $\mathscr{K} \subseteq \mathscr{H}$. A modal formula φ *defines* the class $\mathscr{K}$ *with respect to* $\mathscr{H}$, if for all $(M,w) \in \mathscr{H}$, we have $(M,w) \models \varphi \Leftrightarrow (M,w) \in \mathscr{K}$. If some formula ψ defines a class $\mathscr{J}$ of pointed $(\Pi,\mathscr{R})$-models with respect to the class of all pointed $(\Pi,\mathscr{R})$-models, we simply say that ψ defines $\mathscr{J}$.

If φ is a *sentence* of FO$(\Pi,\mathscr{R})$ and M a $(\Pi,\mathscr{R})$-model, we write $M \models \varphi$ if $(M,f) \models \varphi$ for some assignment f. (Trivially, whether $(M,f) \models \varphi$ holds or not, does not depend on f when φ is a sentence.) If T is a theory consisting of FO$(\Pi,\mathscr{R})$-sentences, we write $M \models T$ iff $M \models \psi$ for all $\psi \in T$. Let $\mathscr{J}$ be a class of *pointed* $(\Pi,\mathscr{R})$-models and T a theory consisting of FO$(\Pi,\mathscr{R})$-*sentences*. We say that the *first-order theory* T *defines the class* $\mathscr{J}$ *of pointed models* if for all pointed $(\Pi,\mathscr{R})$-models (M,w), we have $M \models T \Leftrightarrow (M,w) \in \mathscr{J}$. Notice indeed that accoring to this convention, if T defines a class $\mathscr{J}$ of pointed models and if w is a point in the domain of M and (M,u) a pointed model in $\mathscr{J}$, then we have $(M,w) \in \mathscr{J}$. If a first-order theory T defines a class $\mathscr{H}$ of pointed models, then we say that $\mathscr{H}$ is *definable* by the first-order theory T. If $\mathscr{H}$ is definable by a theory $\{\varphi\}$ containing a single first-order $(\Pi,\mathscr{R})$-sentence φ, we say that $\mathscr{H}$ is definable by the first-order sentence φ.

Let Π and $\mathscr{R} = \{R_1,...,R_k\}$ be finite sets of unary and binary relation symbols, respectively. A *message passing automaton* A over the signature $(\Pi,\mathscr{R})$, or a $(\Pi,\mathscr{R})$-automaton, is a tuple

$$(Q,\mathscr{M},\pi,\delta,\mu,F,G)$$

defined as follows. Q is a nonempty set of *states*. Q can be finite or countably infinite. $\mathscr{M}$ is a nonempty set of *messages*. $\mathscr{M}$ can be finite or countably infinite. For a set S, we let $Pow(S)$ denote the power set of S. $\pi : Pow(\Pi) \to Q$ is an *initial transition function* that determines the beginning state of the automaton A. $\delta : ((Pow(\mathscr{M}))^k \times Q) \to Q$ is a *transition function* that constructs a new state $q \in Q$ when given a k-tuple $(N_1,...,N_k) \in (Pow(\mathscr{M}))^k$ of received message sets and the current state. $\mu : (Q \times \mathscr{R}) \to \mathscr{M}$ is a *message construction function* that constructs a message for the automaton to send forward when given the current state of the automaton and a *communication channel* $R_i \in \mathscr{R}$. $F \subseteq Q$ is the set of *accepting states* of the automaton. $G \subseteq Q \setminus F$ is the set of *rejecting states* of the automaton.

Let $\mathscr{R} = \{R_1,...,R_k\}$ and $\Pi = \{P_1,...,P_m\}$. Let (M,w) be a $(\Pi,\mathscr{R})$-model. The set of R_i-*predecessors* of w is the set of nodes u in the domain of M such that $R_i(u,w)$, and the set of R_i-*successors* of w is the set of nodes u such that $R_i(w,u)$. The set of R_i-successors of w is denoted by $succ(R_i,w)$.

A message passing $(\Pi,\mathscr{R})$-automaton A is *run* on a $(\Pi,\mathscr{R})$-model $M = \big(W,R_1,...,R_k,P_1,...,P_m\big)$, considered to be a distributed system. We first give an intuitive description of the computation of the distributed system defined by the automaton A and the model M, and then define the computation procedure more formally.

On the intuitive level, we place a copy (A,w) of the automaton A to each node $w \in W$. Then, each automaton (A,w) first scans the *local information* of the node w, i.e., finds the set of unary relation symbols $P_i \in \Pi$ such that $(M,w) \models P_i$, and then makes a transition to a *beginning state* based on the local information. The local information at w can be considered to be an m-bit string t of zeros and ones such that the i-th bit of t is 1 iff $(M,w) \models P_i$. After scanning the local information, the automata (A,w), where $w \in W$, begin running in *synchronized steps*. During each step, each automaton (A,w) sends, for each $i \in \{1,...,k\}$, a message m_i to the R_i-*predecessors* of w.[2] The automaton (A,w) also receives a

[2]Therefore information flows opposite to the direction of the arrows (i.e., ordered pairs) of R_i. The reason for this choice is technical, and could be avoided. The choice is due to the relationship between modal logic and message passing automata. A

tuple $(N_1,...,N_k)$ of message sets N_i such that set N_i is received from the R_i-successors of w. Then the automaton updates its state based on the received messages and the current state.

More formally, a $(\Pi,\mathscr{R})$-model $\big(W,R_1,...,R_k,P_1,\ ...,P_m\big)$ and a $(\Pi,\mathscr{R})$-automaton

$$A \ := \ \big(Q,\mathscr{M},\pi,\delta,\mu,F,G\big)$$

define a synchronized distributed computation system which executes *communication rounds* defined as follows. Each round $n \in \mathbb{N}$ defines a *global configuration* $f_n : W \to Q$. The configuration f_0 of the zeroth round is the function f_0 such that $f_0(w) = \pi(\{\ P \in \Pi \mid w \in P^M\ \})$ for all $w \in W$. Recursively, assume that we have defined f_n, and let $(N_1,...,N_k)$ be a tuple of message sets

$$N_i \ = \ \big\{\ m \in \mathscr{M} \mid m = \mu(f_n(v),R_i),\ v \in succ(R_i,w)\ \big\}.$$

Then $f_{n+1}(w) \ = \ \delta\big((N_1,...,N_k),\ f_n(w)\big)$.

When we talk about *the state of the automaton A at the node w in round n*, we mean the state $f_n(w)$. We define that an automaton A *accepts* a pointed model (M,w) if there exists some $n \in \mathbb{N}$ such that $f_n(w) \in F$, and furthermore, for all $m < n$, $f_m(w) \notin G$. Similarly, A *rejects* (M,w) if there exists some $n \in \mathbb{N}$ such that $f_n(w) \in G$, and for all $m < n$, $f_m(w) \notin F$. Notice that A may keep passing messages and changing state even after it has accepted or rejected. Automata that stop sending messages after accepting or rejecting can be modelled in this framework by automata that begin sending only the message "I have halted" once they have accepted or rejected. (Notice that the behaviour of the distributed system does not have to be Turing computable in any sense.)

Let $\mathscr{C}$ be the class of all pointed $(\Pi,\mathscr{R})$-models. Let $\mathscr{H} \subseteq \mathscr{C}$. We say that A accepts (rejects) $\mathscr{H}$ if the class of pointed models in $\mathscr{C}$ that A accepts (rejects) is $\mathscr{H}$. Let $\mathscr{J} \subseteq \mathscr{K} \subseteq \mathscr{C}$. We say that A accepts (rejects) $\mathscr{J}$ in $\mathscr{K}$ if the class of pointed models in $\mathscr{K}$ that A accepts (rejects) is $\mathscr{J}$. A $(\Pi,\mathscr{R})$-automaton A *converges* in the class $\mathscr{K}$ if for all $(M,w) \in \mathscr{K}$, the automaton A either accepts or rejects (M,w). A $(\Pi,\mathscr{R})$-automaton $A = \big(Q,\mathscr{M},\pi,\delta,\mu,F,G\big)$ *halts* in $\mathscr{K}$ if A converges in $\mathscr{K}$, and furthermore, for each state $q \in F \cup G$ that is obtained by A at some $(M,w) \in \mathscr{K}$, the state of A at (M,w) will be q forever once q has been obtained for the first time. We say that the automaton A *specifies a local algorithm* in $\mathscr{K}$ if there exists some $n \in \mathbb{N}$ such that for all $(M,w) \in \mathscr{K}$, the automaton A accepts or rejects (M,w) in some round $m \leq n$. The smallest such number n is called the *effective running time* of A in $\mathscr{K}$. For the sake of curiosity, note that even if A specifies a local algorithm, it does not necessarily halt. However, a corresponding halting automaton of course exists.

Let $\mathscr{K}$ be a class of pointed $(\Pi,\mathscr{R})$-models. When we say that an algorithm A (or more rigorously, a $(\Pi,\mathscr{R})$-automaton A) is *strongly nonlocal* in $\mathscr{K}$, we mean that there exists no $(\Pi,\mathscr{R})$-automaton B that specifies a local algorithm in $\mathscr{K}$ and accepts exactly the the same pointed models in $\mathscr{K}$ as A.

Our framework with $(\Pi,\mathscr{R})$-automata operating on $(\Pi,\mathscr{R})$-models is rather flexible and general. For example, each system in the comprehensive collection of *weak models of distributed computing* studied in [8, 9] can be directly simulated in our framework by restricting attention to suitable classes of $(\Pi,\mathscr{R})$-structures and $(\Pi,\mathscr{R})$-automata. Let us have a closer look at this matter.

Let $\mathscr{R} = \{R\}$ and let Π be any finite set. If M is $(\Pi,\mathscr{R})$-model, where R^M is a symmetric and irreflexive binary relation, then M is an SB(Π)-*model*. The letter S stands for the word *set* and the letter B for *broadcast*. The intuition behind the framework provided by SB(Π)-models is that message passing automata see *neither input port numbers nor output port numbers*. This means that the state transition

possible alternative approach would be to consider modal logics with the truth of $\langle R_i\rangle\varphi$ defined such that $(M,w) \models \langle R_i\rangle\varphi$ iff $\exists v \in W\big((v,w) \in R_i^M$ and $(M,v) \models \varphi\big)$.

of an automaton depends only on the current state and the *set* of messages received—rather than the multiset for example—and an automaton must *broadcast* the same message to *each* of its neighbours during a communication round. It is not possible to send different messages to different neighbours during the same communication round.

The framework provided by SB(Π)-models is similar to the weakest (in computational capacity) computation model SB studied in [8, 9]. In fact, the framework of SB(Π)-models in the current paper is a canonical generalization of the model SB in [8, 9]. In the article [8, 9], all *classes* of structures studied are always associated with a finite maximum degree bound, and furthermore, all structures are assumed to be finite. In the current article, such restrictions need not apply. Also, we allow arbitrary interpretations of the unary relation symbols in Π, while in the SB model of [8, 9], unary relation symbols always indicate the degree of a node in a network (and nothing else).[3] The reason for generalizing the definition of [8, 9] is that in the current paper we opt for generality as well as increased mathematical simplicity. The philosophy in [8, 9] is more application oriented.

Let $n \in \mathbb{N} \setminus \{0\}$ and $S = \{1, ..., n\}$. Let $\Pi = \{P_0, ..., P_n\}$ and $\mathscr{R} = \{R_{(i,j)} \mid (i,j) \in S \times S\}$. A pointed $(\Pi, \mathscr{R})$-model (M, w) is an *n-port-numbering structure*, or a PN(n)-structure, if it satisfies the following (admittedly long and technical, and for the current paper rather unimportant) list of conditions.

1. The union R of the relations $R^M_{(i,j)}$ is a symmetric and irreflexive relation.
2. For any two distinct pairs $(i,j), (k,l) \in S \times S$, if $R^M_{(i,j)}(u,v)$, then $R^M_{(k,l)}(u,v)$ does not hold.
3. For each $(i,j) \in S \times S$, if $R^M_{(i,j)}(u,v)$, then $R^M_{(j,i)}(v,u)$.
4. For each $(i,j) \in S \times S$, the out-degree and in-degree of $R^M_{(i,j)}$ is at most one at each node.
5. If $R_{(i,j)}(u,v)$ for some nodes u and v and some $i, j \in S$, then, if $k < i$, there exists some $l \in S$ and some node v' such that $R^M_{(k,l)}(u,v')$.
6. Similarly, if $R^M_{(i,j)}(u,v)$ for some nodes u and v and some $i, j \in S$, then, if $k < j$, there exists some $l \in S$ and some node u' such that $R^M_{(l,k)}(u',v)$.
7. Finally, for each node u and each $i \in \{0, ..., n\}$, we have $u \in P_i^M$ if and only if the out-degree (or equivalently, in-degree) of the union R of all the relations in $\mathscr{R}$ is i at u.

It is straightforward to show that there exists a first-order $(\Pi, \mathscr{R})$-sentence $\varphi_{\mathrm{PN}(n)}$ that defines the class $\mathscr{PN}(n)$ of all PN(n)-structures. This piece of information will be used in the very end of the current article when we discuss concrete applications of Theorem 4.3. The class of *finite* PN(n)-structures is *exactly* the collection of communication networks of maximum degree at most n used in the framework of the port-numbering model $\mathrm{VV_c}$ of [8, 9]. The related collection of $\mathrm{VV_c}$*-algorithms* corresponds to the class of algorithms that can be specified by $(\Pi, \mathscr{R})$-automata that halt in all finite PN(n)-structures. Therefore the class $\mathscr{PN}(n)$ of exactly all PN(n)-structures, together with $(\Pi, \mathscr{R})$-automata, defines a generalization of the port-numbering model to the context with infinite structures in addition to finite ones. Theorem 4.3 shows that all halting algorithms for $\mathscr{PN}(n)$ are constant-time algorithms. There are no nonlocal halting algorithms in the framework of the port-numbering model when infinite structures are included in the picture.

The port numbering model $\mathrm{VV_c}$ has been studied extensively since the 1980s. The related investigations were originally initiated by Angluin in [2]. Section 3 of [9] gives a brief and accessible introduction to the port-numbering model and its relation to other models of distributed computing.

[3] We do not need the to define the SB model used in [8, 9] for the purposes of the current article. For the precise definition, see [8, 9]. It is worth mentioning here once more, however, that all systems studied in [8, 9] can be directly simulated in our framework by simply restricting attention to suitable automata and suitable classes of pointed models.

3 Halting in the Finite

In this section we prove that when attention is restricted to finite structures, halting and strongly nonlocal algorithms exist even when the model of computing is defined by SB(Π)-models. While the existence of such algorithms may not be surprising, it is by no means a trivial matter. Indeed, as we shall see in Section 4, no such algorithms exist when infinite structures are included in the picture.

Let $\Pi = \{P_0, P_1, Q_1, Q_2, Q_3\}$ and $\mathscr{R} = \{R\}$. We will show that there exists a strongly nonlocal algorithm that halts in the class of finite SB(Π)-models.

We begin by sketching a *rough* intuitive description of the algorithm. The unary relation symbols P_0 and P_1 will be used in order to define binary words in $\{0,1\}^*$ that correspond to *finite walks* in $(\Pi, \mathscr{R})$-models.[4] Each pair (A,w), where A is an automaton and w a node, will store a dynamically growing set of increasingly long finite binary words that correspond to walks that originate from w. The walks will be oriented by the relation symbols Q_1, Q_2 and Q_3 such that if a node u is labelled by Q_i, then its successor is labelled by $Q_{p(i)}$, where $p : \{1,2,3\} \to \{1,2,3\}$ is the cyclic permutation $1 \mapsto 2 \mapsto 3 \mapsto 1$. A pair (A,w) will halt if it records some word $s \in \{0,1\}^*$ that contains a *cube* as a factor, i.e., a word $s = tuuuv$, where u is a *nonempty* word in $\{0,1\}^*$ and $t, v \in \{0,1\}^*$.

Upon halting, (A,w) will send an instruction to halt to its neighbours, who will then pass the message on and also halt. Thus the halting instruction will spread out in the connected component of w, causing further nodes to halt. In addition to detecting a word with a cube factor, a globally spreading halting instruction can also be generated due to the detection of an undesirable labelling pattern defined by the unary predicates in Π. For example, if a node w satisfies both predicates P_0 and P_1, then the labelling pattern at w is undesirable. The intuition is that then w does not uniquely specify an alphabet in $\{0,1\}$, and thereby destroys our intended labelling scheme. Similarly, a halting instruction is generated if a violation of the cyclic permutation scheme of the predicates Q_1, Q_2, Q_3 is detected.

A node accepts iff it halts in a round $n \in \mathbb{N}$ for some positive even number n. Otherwise it rejects upon halting. We shall see that the algorithm is halting and strongly nonlocal in the finite. Strong nonlocality will follow from the existence of arbitrarily long cube-free finite words. Indeed, there exists an infinite cube-free word, known as the *Thue-Morse sequence* (see [1] for example).

We then define the algorithm formally. Let us say that a node w is a Q_1-*node* if $(M,w) \models Q_1 \wedge \neg Q_2 \wedge \neg Q_3$. Similarly, w is a Q_2-node if $(M,w) \models Q_2 \wedge \neg Q_1 \wedge \neg Q_3$ and a Q_3-*node* if $(M,w) \models Q_3 \wedge \neg Q_1 \wedge \neg Q_2$. A node w is *properly oriented* if w is a Q_i-node for some $i \in \{1,2,3\}$, and furthermore, w has a Q_j-node as a neighbour if and only if $j \in \{1,2,3\} \setminus \{i\}$. A node w is *properly labelled* if it is properly oriented, and furthermore, either $(M,w) \models P_0 \wedge \neg P_1$ or $(M,w) \models P_1 \wedge \neg P_0$.

Let $\{0,1\}^+$ denote the set $\{0,1\}^* \setminus \{\lambda\}$, where λ is the empty word. Let $\mathscr{L}$ be the set of finite subsets of $\{0,1\}^+$. The set of states of the automaton A that defines our algorithm is the set

$$\mathbb{S} := \mathscr{L} \times \{0,1\} \times \{Q_1, Q_2, Q_3\} \times \{run, halt\} \times \{0,1\}$$

of quintuples, together with an extra finite set H of *auxiliary states*. The set of messages is

$$\mathbb{M} := \mathscr{L} \times \{1,2,3\} \times \{run, halt\}$$

of triples, together with an additional finite set H' of *auxiliary messages*.

[4] A finite walk in a $(\Pi, \mathscr{R})$-model M is a function from some initial segment of $\mathbb{N}$ into the domain of M such that $\big(f(i), f(i+1)\big) \in R^M$ for each pair $(i, i+1)$ of indices in the domain of f. A finite word $s_0 \ldots s_k = s \in \{0,1\}^*$ *corresponds* to a walk f iff we have $f(i) \in P_{s_i}^M$ for each $i \in \{0, \ldots, k\}$.

We next discuss the intuition behind the definition of the states in $\mathbb{S}$. The first set S_1 of a state $(S_1,S_2,S_3,S_4,S_5) \in \mathbb{S}$ of a node w in round n encodes a collection of words corresponding to walks originating from w. The longer the automaton computes, the longer the words in S_1 get.

The second and third sets S_2 and S_3 are used in order to be able to detect nodes that are not properly labelled. The second set S_2 (intuitively) encodes the symbol $P \in \{P_0,P_1\}$ satisfied by the node w: assuming that the labelling scheme at w is fixed such that $(M,w) \models P_0 \wedge \neg P_1$ or $(M,w) \models P_1 \wedge \neg P_0$, then we have $S_2 = i$ iff w satisfies P_i. Similarly, the third set S_3 intuitively encodes the symbol $Q \in \{Q_1,Q_2,Q_3\}$ such that $(M,w) \models Q$.

The fourth and fifth sets S_4 and S_5 control the halting of the node w. A state (S_1,S_2,S_3,S_4,S_5) is an accepting final state if $S_4 = \mathit{halt}$ and $S_5 = 0$, and rejecting final state if $S_4 = \mathit{halt}$ and $S_5 = 1$. The state $S_5 \in \{0,1\}$ simply counts whether the current computation step is even or odd.

The set S_1 of a message (S_1,S_2,S_3) is a set of binary words. S_1 corresponds to the language recorded by the sending node. S_2 encodes the label in $\{Q_1,Q_2,Q_3\}$ that labels the sending node. S_3 is a halting instruction if $S_3 = \mathit{halt}$.

In the very beginning of the computation, the algorithm makes use of the additional states in H and messages in H' in order to locally detect nodes that are not properly labelled. (It is of course possible that such nodes do not exist.) Then, if a node w is proper and $(M,w) \models P_x \wedge Q_y$, where $x \in \{0,1\}$ and $y \in \{1,2,3\}$, the state of A at w in round 1 is set to be $(\{x\},x,y,\mathit{run},1)$. If w is not proper, then the state of A at w in round 1 is set to be $(\{x'\},x',y',\mathit{halt},1)$, where x' and y' are fixed arbitrarily.

Let U be the set of messages received by a node w in some round $n+1$, where $n \in \mathbb{N} \setminus \{0\}$. Let (S_1,S_2,S_3,S_4,S_5) be the state of w in round n. If $S_4 = \mathit{halt}$, then the new state is the same state (S_1,S_2,S_3,S_4,S_5). Otherwise the new state $(S'_1,S'_2,S'_3,S'_4,S'_5)$ is defined as follows.

Let $p : \{1,2,3\} \to \{1,2,3\}$ be the cyclic permutation $1 \mapsto 2 \mapsto 3 \mapsto 1$. Assume first that U does not contain a tuple of the form (X,Y,halt). Then we define

$$S'_1 = \{\, v \in \{0,1\}^* \mid v = xu \text{ such that } x = S_2 \text{ and } u \in T \text{ for some } (T, p(S_3), \mathit{run}) \in U \,\}.$$

We set $S'_2 = S_2$ and $S'_3 = S_3$. We let $S'_4 = \mathit{halt}$ iff S'_1 contains a word with a cube as a factor. We let $S'_5 \in \{0,1\} \setminus \{S_5\}$.

If U contains a tuple of the form (X,Y,halt), we define $(S'_1,S'_2,S'_3,S'_4,S'_5) = (X',Y',Z,\mathit{halt},x)$, where $x \in \{0,1\} \setminus \{S_5\}$, and X', Y' and Z are fixed arbitrarily.

Let (S_1,S_2,S_3,S_4,S_5) be the state of A at w in round n, where $n \in \mathbb{N} \setminus \{0\}$. If $S_4 = \mathit{run}$, the message broadcast by A at w in round $n+1$ is (S_1,S_3,run), and if $S_4 = \mathit{halt}$, the message is (X,Y,halt), where X and Y are fixed arbitrarily.

Recall that the automaton A accepts iff it halts in round n for some positive even number n. The set of accepting states of the automaton A is exactly the set of states of the type $(X_1,X_2,X_3,\mathit{halt},0)$. The set of rejecting states is the set of states of the type $(X_1,X_2,X_3,\mathit{halt},1)$.

Theorem 3.1. *Let* Π *be as defined above. There exists an* $\mathrm{SB}(\Pi)$ *automaton A that is halting but strongly nonlocal in the class of finite pointed* $\mathrm{SB}(\Pi)$*-models.*

Proof. We shall first establish that the algorithm defined above halts in the class of finite pointed $\mathrm{SB}(\Pi)$-models. Assume that it does not halt in some finite model (M,w). Thus w must be a proper node. By symmetry, we may assume that $(M,w) \models Q_1 \wedge \neg Q_2 \wedge \neg Q_3$. It is easy to see that for each $n \in \mathbb{Z}_+$, the node w must be the first member w_1 of some finite walk $(w_i)_{i \in \{1,\ldots,n\}}$ of proper nodes that satisfy the predicates Q_i in the cyclic fashion such that $(M,w_1) \models Q_1$, $(M,w_2) \models Q_2$, $(M,w_3) \models Q_3$, $(M,w_4) \models Q_1$, and so on. Therefore, since M is a finite model, the node w must be the first member w_1 of some infinite

walk $(w_i)_{i\in\mathbb{Z}_+}$ of proper nodes that satisfy the predicates Q_i in the cyclic fashion. The infinite walk must contain a cycle. The cycle will generate a word with a cube factor that will ultimately be detected at w. Therefore the automaton at w halts. This is a contradiction.

To see that the automaton is strongly nonlocal, we shall consider labelled *path graphs* that encode finite prefixes of the infinite Thue-Morse sequence. The labelled path graphs are defined as follows.

Let ω denote the infinite Thue-Morse sequence of zeros and ones. The sequence does not contain a cube factor. For each finite nonempty prefix ν of ω, let $Path(\nu)$ denote the $(\{P_0,P_1,Q_1,Q_2,Q_3\},\{R\})$-model M such that the following conditions hold.

1. Note first that ν is a nonempty prefix of ω, so ν is a function $\nu:\{0,...,k\}\to\{0,1\}$ for some $k\in\mathbb{N}$. The domain of the model M is the set $\{0,...,k\}$.
2. The model M encodes a path graph, so for each $i,j\in\{0,...,k\}$, we have $(i,j)\in R^M$ iff $|i-j|=1$.
3. M encodes the finite prefix ν of the Thue-Morse sequence, so the following conditions hold for each $i\in\{0,...,k\}$.
   (a) We have $i\in P_0^M$ iff $\nu(i)=0$.
   (b) Similarly, we have $i\in P_1^M$ iff $\nu(i)=1$.
4. Let $j\in\{1,2,3\}$. For each $i\in\{0,...,k\}$, we have $i\in Q_j$ iff $i=j-1 \bmod 3$. Thus we observe that each node of M, with the exception of the end nodes 0 and k, is properly labelled. (Recall the definition of proper labelling from the beginning of this section.)

Since there exist structures $Path(\nu)$ of arbitrarily large finite sizes, and since the Thue-Morse sequence is cube-free, it is easy to see that our automaton A is strongly nonlocal. To see this, assume that there exists a automaton A' such that A' and A accept (and reject) exactly the same finite pointed $\mathrm{SB}(\Pi)$-models, and furthermore, A' specifies a local algorithm. Let $n\in\mathbb{N}$ be the effective running time of A' in the class of finite pointed $\mathrm{SB}(\Pi)$-models. (We assume, w.l.o.g., that n is greater than, say, 10.) Define the prefixes $\nu:\{0,...,5n\}\to\{0,1\}$ and $\nu':\{0,...,5n+1\}\to\{0,1\}$ of the Thue-Morse sequence. Define the pointed models $\big(Path(\nu),\nu(3n)\big)$ and $\big(Path(\nu'),\nu'(3n)\big)$.

Consider the behaviour of our original automaton A on these two pointed models. We claim that A accepts exactly one of the two pointed models. The halting of the pair $(A,\nu(3n))$ in the model $Path(\nu)$ is caused by detecting a violation of the proper labelling scheme at the end point $5n$. Similarly, the halting of $(A,\nu'(3n))$ in the model $Path(\nu')$ is caused by detecting a violation at the end point $5n+1$. The distance from the node $\nu'(3n)$ to the node $\nu'(5n+1)$ is exactly one step greater than the distance from the node $\nu(3n)$ to the node $\nu(5n)$. Thus A accepts exactly one of the pointed models $\big(Path(\nu),\nu(3n)\big)$ and $\big(Path(\nu'),\nu'(3n)\big)$.

Since the pointed models $\big(Path(\nu),\nu(3n)\big)$ and $\big(Path(\nu'),\nu'(3n)\big)$ look locally similar to the automaton A', whose effective running time is n, the automaton A' cannot differentiate between them. Thus A' does not halt on exactly the same finite pointed $\mathrm{SB}(\Pi)$-models as A. This is a contradiction. □

4 Halting and Convergence in Arbitrary Networks

In this section we study a comprehensive collection of distributed computing models in a setting that involves infinite networks in addition to finite ones. We establish that *every* halting distributed algorithm is in fact a local algorithm. In fact, we show that this result relativises to any class of networks definable by a first-order theory.

The strategy of proof in this section is to first appropriately characterize acceptance and rejection of automata in terms of definability in modal logic (see Lemma 4.1), and then use the compactness theorem

in order to obtain the desired end result (see the proof of Theorem 4.3). The characterizations we obtain extend the characterizations in [11].

Let Π be a finite set of unary relation symbols, and let $\mathscr{R} = \{R_1, ..., R_k\}$ be a finite set binary relation symbols. The set T_0 of $(\Pi, \mathscr{R}, 0)$-*types* is defined to be the set containing a conjunction

$$\bigwedge_{P \in U} P \wedge \bigwedge_{P \in \Pi \setminus U} \neg P$$

for each set $U \subseteq \Pi$, and no other formulae. We assume some standard bracketing and ordering of conjuncts, so that there is exactly one conjunction for each set $U \subseteq \Pi$ in T_0. Note also that $\bigwedge \emptyset = \top$. The $(\Pi, \mathscr{R}, 0)$-type $\tau_{(M,w),0}$ of a pointed $(\Pi, \mathscr{R})$-model (M, w) is the unique formula φ in T_0 such that $(M, w) \models \varphi$.

Assume then, recursively, that we have defined the set T_n of $(\Pi, \mathscr{R}, n)$-types. Assume that T_n is finite, and assume also that each pointed $(\Pi, \mathscr{R})$-model (M, w) satisfies *exactly one* type in T_n. We denote this unique type by $\tau_{(M,w),n}$. Define

$$\tau_{(M,w),n+1} := \tau_{(M,w),n} \wedge \bigwedge\{ \langle R_i \rangle \tau \mid \tau \in T_n,\ (M, w) \models \langle R_i \rangle \tau,\ i \in \{1, ..., k\} \} \wedge \bigwedge\{ \neg \langle R_i \rangle \tau \mid \tau \in T_n,\ (M, w) \not\models \langle R_i \rangle \tau,\ i \in \{1, ..., k\} \}.$$

The formula $\tau_{(M,w),n+1}$ is the $(\Pi, \mathscr{R}, n+1)$-type of (M, w). We assume some standard ordering of conjuncts and bracketing, so that if two types $\tau_{(M,w),n+1}$ and $\tau_{(N,v),n+1}$ are equivalent, they are actually the same formula. We define T_{n+1} to be the set

$$\{ \tau_{(M,w),n+1} \mid (M, w) \text{ is a pointed } (\Pi, \mathscr{R})\text{-model} \}.$$

We observe that the set T_{n+1} is finite, and that for each pointed $(\Pi, \mathscr{R})$-model (M, w), there exists exactly one type $\tau \in T_{n+1}$ such that $(M, w) \models \tau$.

It is easy to show by a simple induction on modal depth that each formula φ of $\mathrm{ML}(\Pi, \mathscr{R})$ is equivalent to the disjunction of exactly all $(\Pi, \mathscr{R}, md(\varphi))$-types τ such that $\tau \models \varphi$. Here $\tau \models \varphi$ means that for all pointed $(\Pi, \mathscr{R})$-models (M, w), we have $(M, w) \models \tau \Rightarrow (M, w) \models \varphi$. (Note that $\bigvee \emptyset = \bot$.)

Define $\mathscr{T} := \{ \tau \mid \tau \text{ is a } (\Pi, \mathscr{R}, n)\text{-type for some } n \in \mathbb{N} \}$. A $(\Pi, \mathscr{R})$-*type automaton* A is a $(\Pi, \mathscr{R})$-automaton whose set of states is $\mathscr{T}$. The set of messages is also the set $\mathscr{T}$. The initial transition function π is defined such that the state of A at (M, w) in round $n = 0$ is the $(\Pi, \mathscr{R}, 0)$-type $\tau_{(M,w),0}$. The state transition funtion δ is defined as follows.

Let $n \in \mathbb{N}$. Let $(N_1, ..., N_k)$ be a sequence of sets N_i of $(\Pi, \mathscr{R}, n)$-types. Let τ_n be a $(\Pi, \mathscr{R}, n)$-type. If there exists a type

$$\tau_{n+1} := \tau_n \wedge \bigwedge\{ \langle R_1 \rangle \tau \mid \tau \in N_1 \} \wedge \bigwedge\{ \neg \langle R_1 \rangle \tau \mid \tau \in T_n \setminus N_1 \} \\ \vdots \\ \bigwedge\{ \langle R_k \rangle \tau \mid \tau \in N_k \} \wedge \bigwedge\{ \neg \langle R_k \rangle \tau \mid \tau \in T_n \setminus N_k \},$$

we define $\delta((N_1, ..., N_k), \tau_n) = \tau_{n+1}$. Otherwise we define $\delta((N_1, ..., N_k), \tau_n)$ arbitrarily. On other kinds of input vectors, δ is also defined arbitrarily.

The message construction function μ is defined such that $\mu(\tau, R_i) = \tau$ for each R_i. The sets of accepting and rejecting states can be defined differently for different type automata. It is easy to see that the state of any type automaton A at (M, w) in round n is τ iff the $(\Pi, \mathscr{R}, n)$-type of (M, w) is τ.

Lemma 4.1. *Let Π and $\mathscr{R} = \{R_1, ..., R_k\}$ be finite sets of unary and binary relation symbols, respectively. Let A be a $(\Pi, \mathscr{R})$-automaton. Let $\mathscr{C}$ be the class of pointed $(\Pi, \mathscr{R})$-models. The class $\mathscr{K} \subseteq \mathscr{C}$ of pointed models accepted by A is definable by a (possibly infinite) disjunction $\bigvee \Phi$ of formulae of $\mathrm{ML}(\Pi, \mathscr{R})$. Also the class $\mathscr{J} \subseteq \mathscr{C}$ of pointed models rejected by A is definable by a (possibly infinite) disjunction $\bigvee \Psi$ of formulae of $\mathrm{ML}(\Pi, \mathscr{R})$. The $(\Pi, \mathscr{R}, n)$-type of a pointed $(\Pi, \mathscr{R})$-model $(M, w) \in \mathscr{C}$ is in Φ iff the automaton A accepts (M, w) in round n. Similarly, the $(\Pi, \mathscr{R}, n)$-type of $(N, v) \in \mathscr{C}$ is in Ψ iff the automaton A rejects (N, v) in round n.*

Proof. Let (M, w) be a pointed $(\Pi, \mathscr{R})$-model. Let B be a $(\Pi, \mathscr{R})$-automaton. Let $n \in \mathbb{N}$. We let $B\big((M, w), n\big)$ denote the state of the automaton B at the node w in round n.

We shall first show that for all $n \in \mathbb{N}$ and all pointed $(\Pi, \mathscr{R})$-models (M, w) and (N, v), if the models (M, w) and (N, v) satisfy exactly the same $(\Pi, \mathscr{R}, n)$-type, then $B\big((M, w), m\big) = B\big((N, v), m\big)$ for each $m \leq n$ and each $(\Pi, \mathscr{R})$-automaton B. We prove the claim by induction on n.

For $n = 0$, the claim holds trivially by definition of the transition function π. Let (M, w) and (N, v) be pointed $(\Pi, \mathscr{R})$-models that satisfy the same $(\Pi, \mathscr{R}, n+1)$-type τ_{n+1}. Let B be a $(\Pi, \mathscr{R})$-automaton and δ the transition function of B. Call $q_n = B\big((M, w), n\big)$ and $q_{n+1} = B\big((M, w), n+1\big)$. Let $N_1, ..., N_k$ be sets of $(\Pi, \mathscr{R}, n)$-types such that τ_{n+1} is the formula

$$\tau_n \wedge \bigwedge\{\, \langle R_1 \rangle \tau \mid \tau \in N_1 \,\} \wedge \bigwedge\{\, \neg \langle R_1 \rangle \tau \mid \tau \in T_n \setminus N_1 \,\}$$
$$\vdots$$
$$\bigwedge\{\, \langle R_k \rangle \tau \mid \tau \in N_k \,\} \wedge \bigwedge\{\, \neg \langle R_k \rangle \tau \mid \tau \in T_n \setminus N_k \,\}.$$

Since the models (M, w) and (N, v) satisfy τ_{n+1}, they must satisfy the $(\Pi, \mathscr{R}, n)$-type τ_n. By the induction hypothesis, we therefore conclude that $B\big((M, w), m\big) = B\big((N, v), m\big)$ for each $m \leq n$. In particular, $B\big((N, v), n\big) = q_n$. We must still show that $B\big((N, v), n+1\big) = q_{n+1}$.

Let us define that if L is the set of exactly all $(\Pi, \mathscr{R}, n)$-types τ such that $(M, w) \models \langle R_i \rangle \tau$, then L is the *set of $(\Pi, \mathscr{R}, n)$-types realized by the R_i-successors of w*.

Let $i \in \{1, ..., k\}$. Since (M, w) and (N, v) satisfy the same $(\Pi, \mathscr{R}, n+1)$-type τ_{n+1}, the set of $(\Pi, \mathscr{R}, n)$-types realized by the R_i-successors of the point w is the same as the set realized by the R_i-successors of v; that set is N_i in both cases. Therefore, by the induction hypothesis, the set of states obtained by the R_i-successors of w in round n is exactly the same as the set of states obtained by the R_i-successors of v in round n. This holds for all $i \in \{1, ..., k\}$. Thus w and v receive exactly the same k-tuple of message sets in round $n+1$. Therefore, since $B\big((N, v), n\big) = B\big((M, w), n\big) = q_n$, we conclude that $B\big((N, v), n+1\big) = q_{n+1}$, as required.

We have now established that if (M, w) an (N, v) satisfy the same $(\Pi, \mathscr{R}, n)$-type, then any automaton B produces the same state at (M, w) and (N, v) in all rounds $m \leq n$. We are ready to complete the proof of the current lemma.

Let A be an arbitrary $(\Pi, \mathscr{R})$-automaton. Let $\mathbb{T}$ denote the set

$$\{\, \tau \mid \tau \text{ is a } (\Pi, \mathscr{R}, n)\text{-type for some } n \in \mathbb{N} \,\}.$$

Let Φ denote the set of exactly all types $\tau \in \mathbb{T}$ such that for some n, the type τ is the $(\Pi, \mathscr{R}, n)$-type of some pointed $(\Pi, \mathscr{R})$-model (M, w), and furthermore, the automaton A accepts (M, w) in round n. Define the (possibly infinite) disjunction $\bigvee \Phi$. We shall establish that for all pointed $(\Pi, \mathscr{R})$-models (M, w), we have $(M, w) \models \bigvee \Phi$ iff A accepts (M, w).

Assume that $(M,w) \models \bigvee \Phi$. Thus $(M,w) \models \tau_n$ for some $(\Pi,\mathscr{R},n)$-type τ_n of some pointed model (M',w') accepted by A in round n. The models (M,w) and (M',w') satisfy the same $(\Pi,\mathscr{R},n)$-type τ_n, and thus A produces exactly the same state at (M,w) and at (M',w') in each round $l \leq n$. Therefore (M,w) must be accepted by A in round n.

Assume that (M,w) is accepted by the automaton A. The pointed model (M,w) is accepted in some round n, and thus the $(\Pi,\mathscr{R},n)$-type of (M,w) is one of the formulae in Φ. Therefore $(M,w) \models \bigvee \Phi$.

We have established that $\bigvee \Phi$ defines the class $\mathscr{K} \subseteq \mathscr{C}$. Let $\mathscr{J} \subseteq \mathscr{C}$ be the class of pointed models rejected by A. Let Ψ be the set of types $\tau \in \mathbb{T}$ such that for some n, the type τ is the $(\Pi,\mathscr{R},n)$-type of some pointed $(\Pi,\mathscr{R})$-model (M,w), and furthermore, the automaton A rejects (M,w) in round n. By an argument practically identical to the one above establishing that $\mathscr{K}$ is definable by $\bigvee \Phi$, one can establish that $\bigvee \Psi$ defines the class $\mathscr{J}$. □

Theorem 4.2 (Compactness Theorem, see for example [7])**.** *Assume T is a set of formulae of* $\mathrm{FO}(\Pi,\mathscr{R})$ *such that for each finite subset T' of T, there exists a $(\Pi,\mathscr{R})$-interpretation (M,f) such that $(M,f) \models T'$. Then there exists a $(\Pi,\mathscr{R})$-interpretation (M',f') such that $(M',f') \models T$.*

It is a well-known immediate consequence of the compactness theorem that if $T \models \varphi$, then there is a finite subset T' of T such that $T' \models \varphi$.

Theorem 4.3. *Let Π and $\mathscr{R}$ be finite sets of unary and binary relation symbols. Let $\mathscr{C}$ be the class of all pointed $(\Pi,\mathscr{R})$-models. Let $\mathscr{H} \subseteq \mathscr{C}$ be a class definable by a first-order $(\Pi,\mathscr{R})$-theory. If a $(\Pi,\mathscr{R})$-automaton converges in $\mathscr{H}$, then it specifies a local algorithm in $\mathscr{H}$.*

Proof. Assume a $(\Pi,\mathscr{R})$-automaton A converges in $\mathscr{H} \neq \emptyset$. Let $\mathscr{K} \subseteq \mathscr{H}$ be the class of pointed models accepted by A in $\mathscr{H}$. By Lemma 4.1, there is a disjunction $\bigvee \Phi$ of types that defines $\mathscr{K}$ with respect to $\mathscr{H}$ and a disjunction $\bigvee \Psi$ of types that defines $\mathscr{H} \setminus \mathscr{K}$ with respect to $\mathscr{H}$. The $(\Pi,\mathscr{R},n)$-type of a pointed $(\Pi,\mathscr{R})$-model $(M,w) \in \mathscr{H}$ is in Φ iff the automaton A accepts (M,w) in round n. Similarly, the $(\Pi,\mathscr{R},n)$-type of $(N,v) \in \mathscr{H}$ is in Ψ iff the automaton A rejects (N,v) in round n.

Let T be a first-order theory that defines the class $\mathscr{H}$. Call $X = \{\ \neg St_x(\varphi) \mid \varphi \in \Phi\ \}$ and $Y = \{\ \neg St_x(\varphi) \mid \varphi \in \Psi\ \}$. Since $\bigvee \Phi$ defines $\mathscr{K}$ with respect to $\mathscr{H}$ and $\bigvee \Psi$ defines $\mathscr{H} \setminus \mathscr{K}$ with respect to $\mathscr{H}$, we have $X \cup Y \cup T \models \bot$. By the compactness theorem, there is a finite set $U \subseteq X \cup Y \cup T$ such that $U \models \bot$. Let $V = U \cap X$ and $W = U \cap Y$. Define $W^* = \{\varphi \in \mathrm{ML}(\Pi,\mathscr{R}) \mid St_x(\varphi) \in W\}$, and define V^*, X^* and Y^* analogously. We shall next establish that $\bigwedge W^*$ defines $\mathscr{K}$ with respect to $\mathscr{H}$.

Assume $(M,w) \in \mathscr{K}$. Thus $(M,w) \models Y^*$, and hence $(M,w) \models \bigwedge W^*$. Assume then that $(N,v) \in \mathscr{H} \setminus \mathscr{K}$. Therefore $(N,v) \models X^*$. Since $(N,v) \in \mathscr{H}$, we have $N \models T$. Now assume, for the sake of contradiction, that $(N,v) \models \bigwedge W^*$. Therefore $(N, f[x \mapsto v]) \models X \cup W \cup T$. Thus $(N, f[x \mapsto v]) \models U$. Since $U \models \bot$, we conclude that $(N, f[x \mapsto v]) \models \bot$. This is a contradiction.

We then establish that $\bigwedge V^*$ defines $\mathscr{H} \setminus \mathscr{K}$ with respect to $\mathscr{H}$. Assume $(M,w) \in \mathscr{H} \setminus \mathscr{K}$. Thus $(M,w) \models X^*$, and hence $(M,w) \models \bigwedge V^*$. Assume then that $(N,v) \in \mathscr{K}$. Therefore $(N,v) \models Y^*$. Since $(N,v) \in \mathscr{H}$, we have $N \models T$. Now assume, for the sake of contradiction, that $(N,v) \models \bigwedge V^*$. Therefore $(N, f[x \mapsto v]) \models V \cup Y \cup T$. Thus $(N, f[x \mapsto v]) \models U$. Since $U \models \bot$, we conclude that $(N, f[x \mapsto v]) \models \bot$. This is a contradiction.

The finite sets V^* and W^* are negations of types. Let Φ' be the set of types whose negations are in V^* and Ψ' the set of types whose negations are in W^*. Notice indeed that $\Phi' \subseteq \Phi$ and $\Psi' \subseteq \Psi$. The disjunction $\bigvee \Phi'$ defines $\mathscr{K}$ with respect to $\mathscr{H}$, and the disjunction $\bigvee \Psi'$ defines $\mathscr{H} \setminus \mathscr{K}$ with respect to $\mathscr{H}$.

Let l be the greatest integer j such that there is a $(\Pi,\mathscr{R},j)$-type in $\Phi' \cup \Psi'$. We claim that for each pointed model (M,w) in $\mathscr{H}$, the automaton A either accepts or rejects (M,w) in some round $m \leq l$. To

see this, let $(N,v) \in \mathscr{K}$. Thus $(N,v) \models \bigvee \Phi'$, and hence $(M,w) \models \tau$ for some $(\Pi,\mathscr{R},i)$-type $\tau \in \Phi'$, where $i \leq l$. Since $\Phi' \subseteq \Phi$, we have $\tau \in \Phi$. As we already stated in the beginning of the proof of the current theorem, the $(\Pi,\mathscr{R},n)$-type of a pointed $(\Pi,\mathscr{R})$-model $(M,w) \in \mathscr{H}$ is in Φ iff the automaton A accepts (M,w) in round n. Thus the fact that $\tau \in \Phi$ implies that (N,v) is accepted in round i by A. A similar argument applies when $(N,v) \in \mathscr{H} \setminus \mathscr{K}$. Therefore A specifies a local algorithm in $\mathscr{H}$. □

As we saw in Section 2, each class $\mathscr{PN}(n)$ is definable by a related first-order sentence $\varphi_{\mathrm{PN}(n)}$. Hence all halting algorithms in the port-numbering model are local algorithms when infinite networks are allowed. In Section 3, we saw that finiteness gives rise to nonlocal halting behaviour. It would be interesting to investigate what kinds of other non-first-order properties (in addition to finiteness) there are that lead to existence of nonlocal halting algorithms.

5 Conclusion

We have shown that a comprehensive variety of models of distributed computing cannot define universally halting nonlocal algorithms when infinite networks are allowed. In contrast, we have shown that in the finite, even very weak models of distributed computing can specify universally halting nonlocal algorithms. Our proof concerning infinite networks nicely demonstrates the potential usefulness of modal logic in investigations concerning distributed computing.

Our work in this article concerned *anonymous networks*, i.e., networks without ID-numbers. This choice was due to the fact that in most natural theoretical frameworks for the modelling of computation in infinite networks, even the reading of local IDs would take infinitely long, and thus synchronized communication using ID-numbers would be impossible. This reasoning still leaves the possibility of investigating asynchronous computation. A natural logical framework that can accomodate ID-numbers can probably be based on some variant of hybrid logic (see [3]). Hybrid logic is an extension of modal logic with *nominals*; nominals are formulae that hold in exactly one node. It remains open at this stage, however, how asynchronicity should be treated. Of course there are numerous possibilities, and different logic-based frameworks for similar investigations exist, but we would like to develop an approach that canonically extends the approach introduced in [8, 9], developed further in [11], and used in the current article.

References

[1] J-P. Allouche & J. O. Shallit (2003): *Automatic Sequences - Theory, Applications, Generalizations*. Cambridge University Press, doi:10.1017/CBO9780511546563.

[2] D. Angluin (1980): *Local and Global Properties in Networks of Processors (Extended Abstract)*. In: *STOC*, pp. 82–93, doi:10.1145/800141.804655.

[3] C. Areces & B. ten Cate (2006): *Hybrid Logics*. In P. Blackburn, F. Wolter & J. van Benthem, editors: *Handbook of Modal Logics*, Elsevier, pp. 821–868, doi:10.1016/S1570-2464(07)80017-6.

[4] P. Blackburn & J. van Benthem (2006): *Modal logic: a Semantic Perspective*. In Frank Wolter Patrick Blackburn, Johan van Benthem, editor: *Handbook of Modal Logic*, Elsevier, pp. 1–82, doi:10.1016/S1570-2464(07)80004-8.

[5] P. Blackburn, J. Benthem & F. Wolter (2006): *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA.

[6] P. Blackburn, M. de Rijke & Y. Venema (2001): *Modal Logic*. Cambridge University Press, New York, NY, USA, doi:10.1017/CBO9781107050884.

[7] H-D. Ebbinghaus, J. Flum & W. Thomas (1994): *Mathematical logic*. Undergraduate texts in mathematics, Springer, doi:10.1007/978-1-4757-2355-7.

[8] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela & J. Virtema (2012): *Weak models of distributed computing, with connections to modal logic*. In: *PODC*, pp. 185–194, doi:10.1145/2332432.2332466.

[9] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela & J. Virtema (2014): *Weak Models of Distributed Computing, with Connections to Modal Logic*. Online first, doi:10.1007/s00446-013-0202-3.

[10] N. Immerman (1999): *Descriptive complexity*. Graduate texts in computer science, Springer, doi:10.1007/978-1-4612-0539-5.

[11] A. Kuusisto (2013): *Modal logic and distributed message passing automata*. In: *CSL*, pp. 452–468, doi:10.4230/LIPIcs.CSL.2013.452.

[12] L. Libkin (2004): *Elements of Finite Model Theory*. Springer, doi:10.1007/978-3-662-07003-1.

[13] J. Suomela (2013): *Survey of local algorithms*. *ACM Comput. Surv.* 45(2), p. 24, doi:10.1145/2431211.2431223.