On the Expressiveness of Asynchronous Cellular Automata

Benedikt Bollig

Lehrstuhl für Informatik II, RWTH Aachen, Germany
bollig@informatik.rwth-aachen.de

Abstract. We show that a slightly extended version of asynchronous cellular automata, relative to any class of pomsets and dags without autoconcurrency, has the same expressive power as the existential fragment of monadic second-order logic. In doing so, we provide a framework that unifies many approaches to modeling distributed systems such as the models of asynchronous trace automata and communicating finite-state machines. As a byproduct, we exhibit classes of pomsets and dags for which the radius of graph acceptors can be reduced to 1.

1 Introduction

Distributed systems usually operate concurrently so that some actions do not depend on the occurrence of another. One possible single behavior of a distributed system can therefore be described naturally and in a compact manner by a partially ordered set, whose elements might depend on one another or be executed in either order, whatever the underlying partial-order relation specifies. A partially ordered set, in turn, can be represented by its covering relation or Hasse diagram, which allows to access pairs of events that may follow each other immediately. Accordingly, automata over partial orders, among them *asynchronous cellular automata* (ACAs), usually process input events along their cover relation in a transition-based manner. ACAs have been introduced originally by Zielonka in the framework of Mazurkiewicz traces [17]. They have been generalized by Droste et al. to run on pomsets without autoconcurrency and could be shown to be expressively equivalent to (existential) monadic second-order logic relative to both CROW-pomsets, which are subject to an axiom that considers concurrent read and exclusive owner write, and the more general k-pomsets [6].

However, there might be characteristics of the execution of a distributed system that cannot be captured by the cover relation of a partial order. Consider a directed acyclic graph whose edges reflect the send and receive of messages between sequential processes. Without any additional information (such as communication is both FIFO and reliable), a message exchange cannot be reconstructed from the underlying partial-order relation, which emerges from the reflexive transitive closure of the edge relation. We come across those structures within the framework of communicating systems. A communicating finite-state machine (CFM), for example, is equipped with FIFO channels for exchanging messages. Apart from the ordering of events executed by one and the same sequential automaton, this implies an additional ordering of events that represent the send and receipt of one message. From the partial-order point of view, however, the relation of such communicating events is rather implicit and depends on semantic considerations. Modeling a behavior as a graph rather than a partial order allows for drawing additional edges to make communicating events visible so that automata can access them directly without going over the communication medium. Based on this observation, Kuske introduced the notion of directed acyclic graphs without autoconcurrency, generalizing the partial-order view. He accordingly extended ACAs and showed emptiness relative to the whole class of directed acyclic graphs without autoconcurrency to be decidable [11].

Nevertheless, ACAs are too weak to admit a general logical characterization in terms of a monadic second-order logic beyond classes such as CROW-pomsets and Mazurkiewicz traces, no matter if those structures are represented just by their partial order or by a directed acyclic graph. This is because ACAs process their actions depending solely on what happened in the past. But it is natural to provide systems with the possibility to make *communication requests*. Executing an action is then accompanied by a condition that eventually demands the occurrence of a suitable communication partner. For example, unless we deal with reliable channels anyway, sending a message might come along with the need for being received. Having in mind lossy channel systems with faulty communication [7, 1], ACAs are not even able to specify the set of communication patterns where any message is received, which, however, should be easily formalizable in an appropriate logical calculus. In fact, we will show that, once ACAs are provided with the means of producing communication requests, they are exactly as expressive as the existential fragment of monadic second-order logic over graphs.

Similarly to ACAs, a *vertex-marking graph automaton*, as introduced in [13], collects states it has already read and thereupon assigns a new state to a common successor vertex. Following the idea of communication requests, an acceptance condition in terms of final states is implicitly given by the requirement that any event has to conform to the type of its state. Vertex-marking graph automata appear as a special case of Thomas' graph acceptors [14, 16], which accept a given graph if it can be tiled consistently with a finite supply of patterns. As, relative to a graph class of bounded degree, any first-order definable property is *locally threshold testable* [9], graph acceptors capture precisely the projections of locally threshold testable languages. In turn, this yields a characterization of graph acceptors in terms of the existential fragment of monadic second-order logic. Hereby, the restriction to patterns of radius 1 means loss of expressivity in general. However, in characterizing ACAs (with types) logically, we exhibit many classes of graphs that are relevant for describing the behavior of distributed systems and where the use of patterns of radius 1 is sufficient. The latter property is shared by the domains of words, traces, trees, and grids [15]. The technique we apply generalizes the one used in [2] to characterize the class of CFMs.

Altogether, we combine the models of asynchronous cellular and vertex-marking graph automata towards the new notion of asynchronous cellular automata with types, which encompasses well-known automata concepts and allows a uniform embedding of many existing models of concurrency. Our main contribution is their characterization in terms of existential monadic-second order logic relative to an arbitrary class of directed acyclic graphs without autoconcurrency, covering the expressivity results of subsumed automata models.

The next section recalls the basic concepts of graphs, (existential) monadic second-order logic, and graph acceptors. Section 3 introduces asynchronous automata with types and studies its expressiveness. Finally, Section 4 provides the link between our work and established automata models.

2 Preliminaries

For this section, we fix an alphabet Σ. A *finite directed acyclic graph* (finite dag) is a pair $(V, \lhd)$ where V is its nonempty finite set of *vertices* and $\lhd \subseteq V \times V$ is the *edge relation* such that $\lhd$ is irreflexive and $\lhd^*$ is a partial-order relation. A *Σ-labeled finite dag* (Σ-dag, for short) is a triple $(V, \lhd, \lambda)$ where $(V, \lhd)$ is a finite dag and λ is a mapping $V \to \Sigma$, which we call the *vertex-labeling function*. The set of Σ-dags is denoted by $\mathbb{DAG}(\Sigma)$. We sometimes write $\leq$ for $\lhd^*$ and abbreviate $\lhd^+$ by $<$. Moreover, for $x, y \in V$, let us write $x \lessdot y$ if both $x < y$ and, for any $z \in V$, $x < z \leq y$ implies $z = y$. Then, $(V, \lessdot)$ and $\lessdot$ are called the *Hasse diagram* of $(V, \lhd)$ and, respectively, the *covering relation* of $\leq$. The degree of some $\mathcal{K} \subseteq \mathbb{DAG}(\Sigma)$ is said to be *bounded* if there is some $B \in \mathbb{N}$ such that, for any $(V, \lhd, \lambda) \in \mathcal{K}$ and any $x \in V$, $|\{y \in V \mid x\ (\lhd \cup \lhd^{-1})\ y\}| \leq B$. As usual, we will identify isomorphic structures in the following.

Monadic Second-Order Logic Formulas from monadic second-order (MSO) logic (over Σ) involve first-order variables $x, y, \ldots$ for vertices and second-order variables $X, Y, X_1, X_2, \ldots$ for sets of vertices. They are built up from the atomic formulas $\lambda(x) = a$ (for $a \in \Sigma$), $x \in X$, $x \lhd y$, and $x = y$ and furthermore allow the connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ as well as the quantifiers $\exists$, $\forall$, which can be applied to either kind of variable. Formulas without free variables, which do not occur within the scope of a quantifier, are called sentences. Given $\mathcal{D} = (V, \lhd, \lambda) \in \mathbb{DAG}(\Sigma)$ and an MSO sentence φ, the validity of the satisfaction relation $\mathcal{D} \models \varphi$ is defined canonically with the understanding that first-order variables range over vertices from V and second-order variables over subsets of V. For a set $\mathcal{K} \subseteq \mathbb{DAG}(\Sigma)$ and an MSO sentence φ, the *language* of φ relative to $\mathcal{K}$, denoted by $L_{\mathcal{K}}(\varphi)$, is the set of Σ-dags $\mathcal{D} \in \mathcal{K}$ with $\mathcal{D} \models \varphi$. The class of subsets of $\mathbb{DAG}(\Sigma)$ that can be defined by some MSO sentence φ relative to $\mathcal{K}$ is denoted by $\mathcal{MSO}_{\mathcal{K}}(\Sigma)$. An important fragment of MSO logic is captured by *existential* MSO (EMSO) formulas, which are of the form $\exists X_1 \ldots \exists X_n \varphi$ where φ does not contain any set quantifier. In many cases, the restriction to EMSO formulas suffices to characterize recognizability in terms of automata, e.g., in the domains of words, trees, and Mazurkiewicz traces. Sometimes, however, we even have to restrict to EMSO formulas not to exceed recognizability, because full MSO logic is too expressive in general. The latter applies, for example, to grids and graphs [12] and MSCs [2]. The class $\mathcal{EMSO}_{\mathcal{K}}(\Sigma)$ is defined canonically.

Graph Acceptors Let R be a natural. Given a Σ-dag $\mathcal{D} = (V, \lhd, \lambda)$ and vertices $x, y \in V$, the *distance* $d_{\mathcal{D}}(x, y)$ from x to y in $\mathcal{D}$ is ∞ if $(x, y) \notin (\lhd \cup \lhd^{-1})^*$ and, otherwise, the minimal natural number k such that there is a sequence $x_0, \ldots, x_k \in V$ with $x_0 = x$, $x_k = y$, and $x_i\ (\lhd \cup \lhd^{-1})\ x_{i+1}$ for each $i \in \{0, \ldots, k-1\}$. Sometimes, if it is clear from the context, we omit the subscript $\mathcal{D}$ just writing $d(x, y)$. An *R-sphere* over Σ is a Σ-dag $s = (V, \lhd, \lambda, \gamma)$ together with a designated *sphere center* $\gamma \in V$ such that, for any $x \in V$, $d(x, \gamma) \leq R$. For a Σ-dag $\mathcal{D} = (V, \lhd, \lambda)$ and $x \in V$, let the *R-sphere of $\mathcal{D}$ around x*, denoted by $R\text{-Sph}(\mathcal{D}, x)$, be given by $(V', \lhd', \lambda', x)$ where $V' = \{x' \in V \mid d_{\mathcal{D}}(x', x) \leq R\}$, $\lhd' = \lhd \cap (V' \times V')$, and λ' is the restriction of λ to V'. Figure 1 (b) shows a 1-sphere over $\{a, b, c, d\}$ (with the rectangle as sphere center). It precisely deals with the 1-sphere of the graph aside (Figure 1 (a)) around the d-labeled vertex.

A graph acceptor [14, 16] works on a graph as follows: it first assigns to each node one of its control states and then checks if the local neighborhood of each node (incorporating the states) corresponds to a pattern from a finite supply of spheres. More precisely, a *graph acceptor* over Σ is a structure $\mathcal{B} = (Q, R, \mathcal{S}, \mathit{Occ})$ where Q is its nonempty finite set of *control states*, $R \in \mathbb{N}$ is the *radius*, $\mathcal{S}$ is a finite set of R-spheres over $\Sigma \times Q$, and Occ is a boolean combination of *conditions* of the form "sphere $s \in \mathcal{S}$ occurs at least $n \in \mathbb{N}$ times". A *run* of $\mathcal{B}$ on a Σ-dag $\mathcal{D} = (V, \lhd, \lambda)$ is a mapping $\rho : V \to Q$ such that, for any $x \in V$, the R-sphere of $(V, \lhd, (\lambda, \rho))$ around x is isomorphic to some $s \in \mathcal{S}$. We call ρ *accepting* if the tiling of $\mathcal{D}$ with spheres from $\mathcal{S}$ satisfies Occ. (In the tiling induced by ρ, sphere $s \in \mathcal{S}$ occurs $|\{x \in V \mid s = R\text{-Sph}((V, \lhd, (\lambda, \rho)), x)\}|$ times.) The *language* of $\mathcal{B}$ relative to a class $\mathcal{K} \subseteq \mathbb{DAG}(\Sigma)$, denoted by $L_{\mathcal{K}}(\mathcal{B})$, is the set of Σ-dags $\mathcal{D} \in \mathcal{K}$ on which there is an accepting run of $\mathcal{B}$. Moreover, we set $\mathcal{GA}_{\mathcal{K}}(\Sigma)$ to be $\{L \subseteq \mathbb{DAG}(\Sigma) \mid L = L_{\mathcal{K}}(\mathcal{B})$ for some graph acceptor $\mathcal{B}$ over $\Sigma\}$.

Theorem 1 ([14, 16]). *For any $\mathcal{K} \subseteq \mathbb{DAG}(\Sigma)$ of bounded degree, $\mathcal{GA}_{\mathcal{K}}(\Sigma) = \mathcal{EMSO}_{\mathcal{K}}(\Sigma)$.*

An interesting class of graph languages is characterized by graph acceptors that restrict to 1-spheres [15]. So let us denote by $1\text{-}\mathcal{GA}_{\mathcal{K}}(\Sigma)$ the class $\{L \subseteq \mathbb{DAG}(\Sigma) \mid L = L_{\mathcal{K}}(\mathcal{B})$ for some graph acceptor $\mathcal{B} = (Q, R, \mathcal{S}, \mathit{Occ})$ over Σ with $R = 1\}$. In general, such restricted graph acceptors are strictly weaker. However, we will identify classes of graph languages that allow to restrict to 1-spheres.

3 $\widetilde{\Sigma}$-dags and Asynchronous Cellular Automata

For the rest of this paper, we fix a nonempty finite set Ag of at least two *agents* and a *distributed alphabet* $\widetilde{\Sigma}$, which is a tuple $(\Sigma_i)_{i \in Ag}$ of (not necessarily disjoint) alphabets such that (for rather technical reasons) $\Sigma_i \not\subseteq \Sigma_j$ for any $i \neq j$. Let in the following Σ stand for $\bigcup_{i \in Ag} \Sigma_i$, the set of *actions*. Elements from Σ_i are understood to be actions that are performed by agent i. So let, for $a \in \Sigma$, $loc(a) := \{i \in Ag \mid a \in \Sigma_i\}$ denote the set of agents that are involved in a. Having this in mind, we say that actions a and b are *independent* and write $a\ I_{\widetilde{\Sigma}}\ b$ if there is no common agent that controls both of them, i.e., if $loc(a) \cap loc(b) = \emptyset$.

(a) (b)

Fig. 1. An $(\{a\},\{b,c\},\{c,d\})$-dag and a 1-sphere

Otherwise, we say a and b are dependent, writing $a \; D_{\widetilde{\Sigma}} \; b$. We now introduce the model representing the behavior of a system of communicating agents. In doing so, we combine the standard models of [6] and [11]:

Definition 1. *A* $\widetilde{\Sigma}$-dag *is a* Σ*-dag* $(V, \lhd, \lambda)$ *such that*

- *for any* $i \in Ag$, $\lambda^{-1}(\Sigma_i)$ *is linearly ordered by* $\leq$ *and*
- *for any* $(x,y),(x',y') \in \lhd$ *with* $\lambda(x) \; D_{\widetilde{\Sigma}} \; \lambda(x')$ *and* $\lambda(y) \; D_{\widetilde{\Sigma}} \; \lambda(y')$, *we have* $x \leq x'$ *iff* $y \leq y'$.

Thus, for any $x \in V$ and $a \in \Sigma$, there is at most one vertex $y \in V$ such that both $x \lhd y$ ($y \lhd x$) and $\lambda(y) = a$. Intuitively, the second condition makes sure that communication between two processes cannot cross. When we consider communicating systems with message exchange, this corresponds to a FIFO architecture where messages (x,y) and (x',y') of equal type are received in terms of y and y' in the order they have been sent in terms of x and x', respectively.

An $(\{a\},\{b,c\},\{c,d\})$-dag is depicted in Figure 1 (a). We denote by $\mathbb{DAG}(\widetilde{\Sigma})$ the set of $\widetilde{\Sigma}$-dags. Note that any $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$ has bounded degree. A useful subclass of $\mathbb{DAG}(\widetilde{\Sigma})$, denoted by $\mathbb{DAG}_H(\widetilde{\Sigma})$, is the set of graphs $(V, \lhd, \lambda) \in \mathbb{DAG}(\widetilde{\Sigma})$ such that $\lhd = \lessdot$, i.e., $(V, \lhd)$ is the Hasse diagram of some partially ordered set. Let $(V, \lhd, \lambda)$ be a $\widetilde{\Sigma}$-dag and let $x \in V$. For $i \in Ag$, we say that x is Σ_i-maximal if $\lambda(x) \in \Sigma_i$ and there is no $y \in \lambda^{-1}(\Sigma_i)$ such that $x < y$. Note that there is at most one Σ_i-maximal vertex. We denote by $\mathrm{R}(x) := \{a \in \Sigma \mid$ there is some $y \in V$ such that $y \lhd x$ and $\lambda(y) = a\}$ the *read domain* of x and, given $a \in \mathrm{R}(x)$, let a-pred(x) be the unique vertex y such that both $y \lhd x$ and $\lambda(y) = a$. Accordingly, let $\mathrm{W}(x) := \{a \in \Sigma \mid$ there is some $y \in V$ such that $x \lhd y$ and $\lambda(y) = a\}$ be the *write domain* of x and, for $a \in \mathrm{W}(x)$, a-succ(x) denote the unique vertex y such that both $x \lhd y$ and $\lambda(y) = a$.

Example 1 (Mazurkiewicz Traces). A *(Mazurkiewicz) trace* [5] over $\widetilde{\Sigma}$ is a $\widetilde{\Sigma}$-dag $(V, \lhd, \lambda) \in \mathbb{DAG}_H(\widetilde{\Sigma})$ such that, for any $x, y \in V$, $x \lhd y$ implies $\lambda(x) \; D_{\widetilde{\Sigma}} \; \lambda(y)$. Note that, as we consider a subclass of $\mathbb{DAG}_H(\widetilde{\Sigma})$, $\lhd$ and $\lessdot$ coincide. The set of traces over $\widetilde{\Sigma}$ is denoted by $\mathbb{TR}(\widetilde{\Sigma})$. The dag from Figure 1 (a) is clearly not a trace over $(\{a\},\{b,c\},\{c,d\})$: neither is it a Hasse diagram, nor are neighboring vertices consistently labeled with dependent actions.

Example 2 (Message Sequence Charts). Messages might be exchanged between the agents of a distributed system by performing send and receive actions. So set, for an agent $i \in Ag$, Γ_i to be $\{i!j \mid j \in Ag \setminus \{i\}\} \cup \{i?j \mid j \in Ag \setminus \{i\}\}$, the set of *(communication) actions* of agent i. The action $i!j$ is to be read as "i sends a message to j", while $j?i$ is the complementary action of receiving a message sent from i to j. Moreover, let Γ stand for the union of the Γ_i and set $\widetilde{\Gamma}$ to be the distributed alphabet $(\Gamma_i)_{i \in Ag}$. A *message sequence chart* (MSC) over Ag is a $\widetilde{\Gamma}$-dag $(V, \lessdot, \lambda)$ such that (i) for any $i \in Ag$, $\lessdot \cap (\lambda^{-1}(\Gamma_i) \times \lambda^{-1}(\Gamma_i))$ is the cover relation of some linear order, (ii) for any $(x, y) \in \lessdot$ satisfying $\lambda(x)\ I_{\widetilde{\Gamma}}\ \lambda(y)$, $\lambda(x)$ is a send action and $\lambda(y)$ is its complementary receive, and (iii) for any $x \in V$, there is $y \in V$ satisfying both $\lambda(x)\ I_{\widetilde{\Gamma}}\ \lambda(y)$ and $x\ (\lessdot \cup \lessdot^{-1})\ y$. We denote by $\mathbb{MSC}(Ag)$ the set of MSCs over Ag. Note that, by the definition of a $\widetilde{\Gamma}$-dag, an MSC behaves in a FIFO manner, neglecting overtaking of messages of equal type. If we do not require a send vertex to be equipped with a corresponding receive, we obtain the class of (potentially) *lossy* MSCs over Ag, which is a superset of $\mathbb{MSC}(Ag)$ and shall be denoted by $\mathbb{LMSC}(Ag)$. If, in Figures 2 (a) and (b), we replace b with $1!2$ and a with $2?1$, we obtain an MSC over $\{1, 2\}$ and, respectively, a lossy MSC over $\{1, 2\}$, which is not an MSC. Note that $\mathbb{MSC}(Ag)$ might be defined relative to $\mathbb{LMSC}(Ag)$ by the (first-order) sentence $\forall x \bigwedge_{i \in Ag,\ j \in Ag \setminus \{i\}} (\lambda(x) = i!j \rightarrow \exists y(x \lessdot y \wedge \lambda(y) = j?i))$.

Definition 2. *An* asynchronous cellular automaton with types (ACAT) *over $\widetilde{\Sigma}$ is a structure (Q, Δ, T, F) where*

- Q *is the nonempty finite set of* states*,*
- $\Delta \subseteq (Q \uplus \{-\})^{\Sigma} \times \Sigma \times Q$ *is the set of* transitions*,*
- $T : (\Sigma \times Q) \rightarrow 2^{\Sigma}$ *is the* type function *such that $b \in T(a, q)$ implies $a\ I_{\widetilde{\Sigma}}\ b$,*
- $F \subseteq (Q \uplus \{\imath\})^{Ag}$ *is the set of* final states*.*

So let $\mathcal{A} = (Q, \Delta, T, F)$ be an ACAT over $\widetilde{\Sigma}$. Note that $\overline{q} \in (Q \uplus \{-\})^{\Sigma}$ can be considered to be a subset of $\Sigma \times Q$ with the understanding that, for any $a \in \Sigma$ and $q \in Q$, $(a, q) \in \overline{q}$ iff $\overline{q}[a] = q$. In the following, we often write a transition $(\overline{q}, a, q) \in \Delta$ as $\overline{q} \rightarrow (a, q)$ with $\overline{q}$ being a subset of $\Sigma \times Q$. Let $\mathcal{D} = (V, \lessdot, \lambda)$ be a $\widetilde{\Sigma}$-dag and r be a mapping $V \rightarrow Q$. We define a mapping $r^- : V \rightarrow (Q \uplus \{-\})^{\Sigma}$ setting $r^-(x)[a]$ to be $-$ if $a \notin \mathrm{R}(x)$ and to be $r(a\text{-pred}(x))$ if $a \in \mathrm{R}(x)$. A *run* of $\mathcal{A}$ on $\mathcal{D}$ is a mapping $r : V \rightarrow Q$ such that, for any $x \in V$, we have $(r^-(x), \lambda(x), r(x)) \in \Delta$. It remains to constitute when r is accepting. For any $i \in Ag$ with $\lambda^{-1}(\Sigma_i) \neq \emptyset$, let f_i be given by $r(x)$ where x is the Σ_i-maximal vertex in V. For any other $i \in Ag$, set f_i to be $\imath$. The run r is *accepting* if both $(f_i)_{i \in Ag} \in F$ and, for any $x \in V$, we have $T(\lambda(x), r(x)) \subseteq \mathrm{W}(x)$. The intuition behind the latter condition is that we require $\mathrm{W}(x)$ to contain at least the communication requests imposed by the type function of the automaton.[1] Given $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, we denote by $L_{\mathcal{K}}(\mathcal{A})$ the *language* of $\mathcal{A}$ relative to $\mathcal{K}$, i.e.,

[1] Note that we could even require $T(\lambda(x), r(x)) = \mathrm{W}(x) \cap \{a \in \Sigma \mid a\ I_{\widetilde{\Sigma}}\ \lambda(x)\}$ without affecting the expressiveness of ACATs.

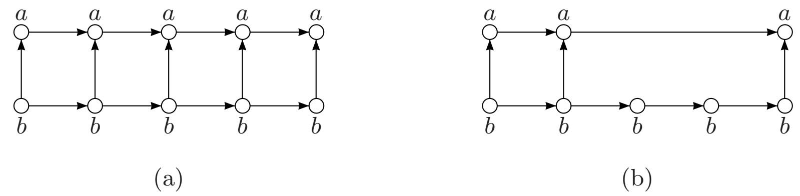

Fig. 2. The $(\{a\},\{b\})$-dags $\mathcal{D}_5$ and $\mathcal{D}_5[2,4]$

the set of $\widetilde{\Sigma}$-dags $\mathcal{D} \in \mathcal{K}$ such that there is an accepting run of $\mathcal{A}$ on $\mathcal{D}$. An ACAT $\mathcal{A} = (Q, \Delta, T, F)$ over $\widetilde{\Sigma}$ is called an *asynchronous cellular automaton* (ACA) over $\widetilde{\Sigma}$ if $T(a,q) = \emptyset$ for any $a \in \Sigma$ and $q \in Q$. For $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, let $\mathcal{ACAT}_{\mathcal{K}}(\widetilde{\Sigma}) = \{L \subseteq \mathbb{DAG}(\widetilde{\Sigma}) \mid L = L_{\mathcal{K}}(\mathcal{A})$ for some ACAT $\mathcal{A}$ over $\widetilde{\Sigma}\}$. The class $\mathcal{ACA}_{\mathcal{K}}(\widetilde{\Sigma})$ is defined accordingly.

Example 3. Let L be the set of $(\{a\},\{b\})$-dags $\mathcal{D}_n = (V_n, \lhd_n, \lambda_n)$, $n \geq 1$, where $V_n = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$, $\lhd_n$ is the union of $\{(x_i, x_{i+1}) \mid i \in \{1, \ldots, n-1\}\}$, $\{(y_i, y_{i+1}) \mid i \in \{1, \ldots, n-1\}\}$, and $\{(y_i, x_i) \mid i \in \{1, \ldots, n\}\}$, and the x_i are labeled by λ_n with a, while the y_i are labeled with b. $\mathcal{D}_5$ is depicted in Figure 2 (a). An ACAT $\mathcal{A}$ with $L_{\mathbb{DAG}((\{a\},\{b\}))}(\mathcal{A}) = L$ is (Q, Δ, T, F) where $Q = \{q_1, q_2\}$ and Δ contains the transitions $\emptyset \to (b, q_2)$, $\{(b, q_2)\} \to (b, q_2)$, $\{(b, q_2)\} \to (a, q_1)$, and $\{(a, q_1), (b, q_2)\} \to (a, q_1)$. Moreover, we set $T(a, q_1) = T(b, q_1) = T(a, q_2) = \emptyset$, $T(b, q_2) = \{a\}$, and $F = \{(q_1, q_2)\}$. An accepting run of $\mathcal{A}$ on $\mathcal{D}_n$ assigns q_1 to any a-labeled vertex and q_2 to any b-labeled one.

Lemma 1 ([6]). $\mathcal{ACA}_{\mathbb{DAG}(\widetilde{\Sigma})}(\widetilde{\Sigma}) \subset \mathcal{ACAT}_{\mathbb{DAG}(\widetilde{\Sigma})}(\widetilde{\Sigma})$

Proof. The language L from Example 3 cannot be recognized by some ACA over $(\{a\},\{b\})$ relative to $\mathbb{DAG}((\{a\},\{b\}))$. For suppose there is an ACA $\mathcal{A}$ over $(\{a\},\{b\})$ with $L_{\mathbb{DAG}((\{a\},\{b\}))}(\mathcal{A}) = L$. For $n \geq 1$, suppose furthermore r to be a run of $\mathcal{A}$ on $\mathcal{D}_n$. If n has been chosen large enough, there are $1 \leq i < j < n$ such that $r(x_i) = r(x_j)$. From $\mathcal{D}_n$, we obtain the $(\{a\},\{b\})$-dag $\mathcal{D}_n[i,j]$ by removing from V_n the vertices $x_{i+1}, \ldots, x_j$ and from $\lhd_n$ any edge touching some of these nodes and by adding instead an edge from x_i to x_{j+1}. Though $\mathcal{D}_n[i,j] \notin L$, $\mathcal{A}$ admits an accepting run on $\mathcal{D}_n[i,j]$. The reason is that, in the example, $\mathcal{A}$ has no means to impose on a b-labeled vertex an a-labeled successor. $\square$

Though ACATs are generally strictly more expressive than ACAs, many distributed systems allow for dropping types: we call a class $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$ *communication closed* if, for any $(V_1, \lhd_1, \lambda_1), (V_2, \lhd_2, \lambda_2) \in \mathcal{K}$, any $a \in \Sigma$, and any $x_1 \in V_1$ and $x_2 \in V_2$ with $\lambda(x_1) = \lambda(x_2) = a$, we have $\mathrm{W}(x_1) \cap I_{\widetilde{\Sigma}}(a) = \mathrm{W}(x_2) \cap I_{\widetilde{\Sigma}}(a)$ (where $I_{\widetilde{\Sigma}}(a)$ shall contain any $b \in \Sigma$ with $a\, I_{\widetilde{\Sigma}}\, b$).

Lemma 2. *Let $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$. If $\mathcal{K}$ is communication closed, then we have $\mathcal{ACA}_{\mathcal{K}}(\widetilde{\Sigma}) = \mathcal{ACAT}_{\mathcal{K}}(\widetilde{\Sigma})$.*

Proof. Let $\mathcal{A} = (Q, \Delta, T, F)$ be an ACAT over $\widetilde{\Sigma}$ and suppose $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$ to be communication closed. Then, for any $a, b \in \Sigma$ with $a \; I_{\widetilde{\Sigma}} \; b$ such that $\mathcal{K}$ requires an a-labeled vertex to be directly followed by some b-labeled vertex, we can, without changing the recognized language relative to $\mathcal{K}$, remove b from any communication request $T(a, q)$. Now suppose $\mathcal{K}$ prevents a from being followed by b. Then, for every $q \in Q$ with $b \in T(a, q)$, we remove any transition $\overline{q} \to (a, q)$ and set $T(a, q) = \emptyset$. It is easily seen that we obtain an ACA, which, moreover, is equivalent to $\mathcal{A}$ relative to $\mathcal{K}$. □

The class $\mathbb{TR}(\widetilde{\Sigma})$ of Mazurkiewicz traces is trivially communication closed, as none pair of neighboring nodes can be labeled with independent actions. The class $\mathbb{MSC}(Ag)$ is communication closed, too: any sending vertex has exactly one successor vertex, labeled with the corresponding receive action, that is not controlled by the same agent. However, the class $\mathbb{LMSC}(Ag)$ of lossy MSCs can no longer do without types, as an easy adaption of the proof of Lemma 1 shows.

Theorem 2. *For any $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, $\mathcal{ACAT}_{\mathcal{K}}(\widetilde{\Sigma}) = \mathcal{EMSO}_{\mathcal{K}}(\Sigma)$. Moreover, both conversions, from automata to formulas and vice versa, are effective.*

The theorem follows from Lemma 3 and Lemma 4, which will be shown below.

Theorem 3. *For any $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, we have $\mathcal{GA}_{\mathcal{K}}(\Sigma) = 1\text{-}\mathcal{GA}_{\mathcal{K}}(\Sigma)$.*

Proof. Let $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$ and let $\mathcal{B}$ be a graph acceptor over Σ. According to Theorems 1 and 2, there is an ACAT $\mathcal{A} = (Q, \Delta, T, F)$ over $\widetilde{\Sigma}$ such that $L_{\mathcal{K}}(\mathcal{A}) = L_{\mathcal{K}}(\mathcal{B})$. Without loss of generality, we shall assume that, for any $\overline{q} \in F$ and $i \in Ag$ with $\overline{q}[i] \in Q$, $\overline{q}[i]$ can be assigned at most to the Σ_i-maximal vertex of a $\widetilde{\Sigma}$-dag. A graph acceptor $\mathcal{B}'$ of radius 1 with $L_{\mathcal{K}}(\mathcal{B}') = L_{\mathcal{K}}(\mathcal{A})$ is given by $(Q, 1, \mathbb{S}, Occ)$ where, for any transition $\{(a_1, p_1), \ldots, (a_n, p_n)\} \to (b, q) \in \Delta$ and any $\{(c_1, q_1), \ldots, (c_m, q_m)\} \subseteq \Sigma \times Q$ with $c_i \neq c_j$ $(i \neq j)$ and $T(b, q) \subseteq \{c_1, \ldots, c_m\}$, $\mathbb{S}$ contains s if removing from s the edges that do not touch its center yields $\{(a_1, p_1), \ldots, (a_n, p_n)\} \to (b, q) \to \{(c_1, q_1), \ldots, (c_m, q_m)\}$ (with the expected meaning). Now let, given $q \in Q$, $\mathbb{S}_q$ contain those spheres whose sphere center is labeled with (a, q) for some a, and let, accordingly, $\mathbb{S}_a$ with $a \in \Sigma$ contain those spheres whose sphere center is labeled with (a, q) for some q. Then, $Occ = \bigvee_{\overline{q} \in F} \left(\bigwedge_{i \in Ag,\ \overline{q}[i] \in Q} \bigvee_{s \in \mathbb{S}_{\overline{q}[i]}} \text{“}s \geq 1\text{”} \wedge \bigwedge_{i \in Ag,\ \overline{q}[i] = \imath,\ a \in \Sigma_i,\ s \in \mathbb{S}_a} \neg\text{“}s \geq 1\text{”} \right)$ guarantees that a run of $\mathcal{B}$ is accepting only if the corresponding run of $\mathcal{A}$ is. □

Lemma 3. *For any $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, $\mathcal{ACAT}_{\mathcal{K}}(\widetilde{\Sigma}) \subseteq \mathcal{EMSO}_{\mathcal{K}}(\Sigma)$.*

Proof. The construction of an EMSO sentence from a given ACAT follows similar instances of that problem. See, for example, [6]. Basically, an interpretation of second-order variables (actually, an interpretation that stands for a partition $(X_q)_{q \in Q}$ of the set of vertices at hand) means an assignment of states to vertices, which is then checked within the first-order fragment of the formula for being an accepting run. Hereby, the type function can be handled by the formula $\forall x \bigwedge_{a \in \Sigma,\ q \in Q} ((\lambda(x) = a \wedge x \in X_q) \to \bigwedge_{b \in T(a,q)} \exists y (x \lessdot y \wedge \lambda(y) = b))$. □

Lemma 4. *For any* $\mathcal{K} \subseteq \mathbb{DAG}(\widetilde{\Sigma})$, $\mathcal{EMSO}_{\mathcal{K}}(\Sigma) \subseteq \mathcal{ACAT}_{\mathcal{K}}(\widetilde{\Sigma})$.

Proof. In the following, we generalize the technique that has been applied to CFMs in [2]. So suppose φ to be an EMSO sentence. According to Theorem 1, there is a graph acceptor $\mathcal{B} = (Q, R, \mathcal{S}, Occ)$ over Σ such that $L_{\mathcal{K}}(\mathcal{B}) = L_{\mathcal{K}}(\varphi)$. We now transform $\mathcal{B}$ into an ACAT $\mathcal{A}$ over $\widetilde{\Sigma}$ such that $L_{\mathcal{K}}(\mathcal{A}) = L_{\mathcal{K}}(\mathcal{B})$.

Let us first give an intuition of how $\mathcal{A}$, which operates rather locally, simulates the global control of $\mathcal{B}$: basically, any state of $\mathcal{A}$ makes a guess about the local environment of radius R that it is about to read and then verifies its guess by passing it through a run and checking if other components have made their guess accordingly. A guess is actually an *extended* R*-sphere* $\sigma = (V, \lhd, \lambda, \gamma, \alpha, i)$ where $core(\sigma) := (V, \lhd, \lambda, \gamma) \in \mathcal{S}$ is the pattern that $\mathcal{A}$ expects to see, $\alpha \in V$ is the *active vertex*, which corresponds to the vertex that $\mathcal{A}$ is about to read, and $i \in \{1, \ldots, const\}$ is the current *instance* of the pattern where $const := (2|\Sigma| + 1) \cdot (\max\{|V| \mid (V, \lhd, \lambda, \gamma) \in \mathcal{S}\})^2$. The ACAT $\mathcal{A}$, reading some vertex x and entering a state associated with a guess σ, presumes that x is to its local environment as α is to $core(\sigma)$. In other words, $\mathcal{A}$ considers x to be the analogon of α and the environment of x to look like $core(\sigma)$. To establish isomorphism between $core(\sigma)$ and the environment around x, $\mathcal{A}$ transfers σ to the immediate successor vertices x' of x except that, in σ, the active vertex α is replaced with some α' such that $\alpha \lhd \alpha'$. This is because x shall correspond to x' only if α corresponds to α'. As, in almost all cases, a tiling by $\mathcal{B}$ induces an overlapping of participating spheres, a state of $\mathcal{A}$ actually holds a *s*et of extended R-spheres, which subsequently have to be forwarded and verified simultaneously. The state entered when reading x carries exactly one extended R-sphere whose sphere center and active vertex coincide with the understanding that the corresponding run of $\mathcal{B}$ assigns to x the control state associated with precisely that sphere center. There may even be an overlapping of isomorphic R-spheres so that a state possibly contains several instances of one and the same sphere, which then refer to distinct vertices as corresponding sphere centers. Those instances will be distinguished by means of the natural i. However, there can be at most *const* such overlappings, an order of magnitude that depends on $\mathcal{B}$ only. A second component of a state keeps track of the number of spheres used so far.

Now set $\mathcal{S}^+$ to be the set of extended R-spheres that emerge from $\mathcal{S}$. In the scope of extended spheres $\sigma, \sigma' \in \mathcal{S}^+$, we let in the following $V, \lhd, \lambda, \gamma, \alpha, i$ and $V', \lhd', \lambda', \gamma', \alpha', i'$ refer to the components of σ and σ', respectively. Given $\sigma \in \mathcal{S}^+$, and $y \in V$, $\sigma[y]$ shall denote the extended R-sphere $(V, \lhd, \lambda, \gamma, y, i) \in \mathcal{S}^+$, in which the active vertex α of σ is replaced with a new active vertex y. Finally, $\max(Occ)$ shall denote the least threshold n such that Occ does not distinguish occurrence numbers $\geq n$. Let us now turn to the construction of the ACAT $\mathcal{A} = (Q', \Delta, T, F)$, which is given as follows: a state of $\mathcal{A}$ is a pair $(\mathcal{S}, \nu)$ where ν is a mapping $\mathcal{S} \to \{0, \ldots, \max(Occ)\}$ and $\mathcal{S}$ is a subset of $\mathcal{S}^+$ such that (i) there is exactly one extended sphere $\sigma \in \mathcal{S}$ with $\gamma = \alpha$ (we set $core(\mathcal{S})$ to be $core(\sigma)$), (ii) there is $(a, q) \in \Sigma \times Q$ such that, for any $\sigma \in \mathcal{S}$, $\lambda(\alpha) = (a, q)$ (so that we can assign a well-defined unique label $\ell(\mathcal{S}) := a$ to $\mathcal{S}$), and (iii) for any $\sigma, \sigma' \in \mathcal{S}$, if $core(\sigma) = core(\sigma')$ and $i = i'$, then $\alpha = \alpha'$.

Let $(\overline{\mathcal{S}}, b, \mathcal{S}') \in \Delta$ if the following hold:

L $\quad \ell(\mathcal{S}') = b$.

C $\quad \mathcal{S}'(s)$ is the minimum of $\max(Occ)$ and

$$\begin{cases} \max(\{0\} \cup \{\overline{\mathcal{S}}[a](s) \mid a \in \Sigma,\ \overline{\mathcal{S}}[a] \neq -\}) & \text{if } s \neq \mathit{core}(\mathcal{S}') \\ \max(\{1\} \cup \{\overline{\mathcal{S}}[a](s) + 1 \mid a \in \Sigma,\ \overline{\mathcal{S}}[a] \neq -\}) & \text{if } s = \mathit{core}(\mathcal{S}') \end{cases}$$

For any $a \in \Sigma$ with $\overline{\mathcal{S}}[a] \neq -$, any $\sigma \in \overline{\mathcal{S}}[a]$, and any $y \in V$:

W1 If $\sigma[y] \in \mathcal{S}'$, then $\alpha \lhd y$.
W2 If $b \notin \mathrm{W}(\alpha)$, then $d(\alpha, \gamma) = R$.
W3 If $b \in \mathrm{W}(\alpha)$, then $\sigma[b\text{-succ}(\alpha)] \in \mathcal{S}'$.

For any $a \in \Sigma$ and any $\sigma \in \mathcal{S}'$:

R1 If $\overline{\mathcal{S}}[a] \neq -$ and $a \notin \mathrm{R}(\alpha)$, then $d(\alpha, \gamma) = R$.
R2 If $a \in \mathrm{R}(\alpha)$, then $\sigma[a\text{-pred}(\alpha)] \in \overline{\mathcal{S}}[a] \neq -$.

Fig. 3. Simulating a graph acceptor

The definition of Δ is given by Figure 3. For ease of notation, we hereby use $\overline{\mathcal{S}}[a]$ and $\mathcal{S}'$ to denote any component of a state, i.e., $\overline{\mathcal{S}}[a]$ and $\mathcal{S}'$ each refer to both a set of extended spheres and a mapping $\mathbb{S} \to \{0, \ldots, \max(Occ)\}$. Conditions **L** and **C** go without saying. In particular, **C** implements a simple threshold counting procedure. Now assume an extended sphere σ with active vertex α is attached to some x. Assume furthermore that $\sigma[y]$ is attached to some direct successor x' of x. As α and y have to simulate x and x' (and vice versa), they have to be joint by an edge as well. This is what condition **W1** is supposed to guarantee. Suppose now that α lacks a b-successor, while x does not. That situation is allowed only if the distance from α to γ is R, as then the scope of σ ends anyway so that, beyond x, it is no longer responsible for what will happen (**W2**). Otherwise, if b is contained in the write domain of α, then σ has to coincide with the input structure further on so that $\sigma[b\text{-succ}(\alpha)]$ is sent to the b-labeled successor of x (**W3**). The duals of **W2** and **W3**, regarding the read domain of a vertex, are guaranteed by conditions **R1** and **R2**, respectively. Note that condition **W1** lacks its dual case, as this is implicitly present.

Let us now turn to the type function and the set of final states of $\mathcal{A}$: for any $a \in \Sigma$ and $(\mathcal{S}, \nu) \in Q'$, we set $T(a, (\mathcal{S}, \nu))$ to be $\{b \in \Sigma \mid b\ I_{\widetilde{\Sigma}}\ a$ and there is $\sigma \in \mathcal{S} :\ b \in \mathrm{W}(\alpha)\}$, i.e., if the active vertex of some extended sphere from $\mathcal{S}$ has some b-labeled successor, then so will the vertex to which $\mathcal{S}$ is attached. Finally, $\overline{q} \in (Q' \uplus \{\imath\})^{Ag}$ shall be contained in F if both, for any $i \in Ag$ and any $\sigma \in \overline{q}[i]$, $\mathrm{W}(\alpha) \cap \Sigma_i = \emptyset$ and the union of mappings that occur in $\overline{q}$ (for each sphere, take the mapping with the maximum occurrence number) satisfies the requirements imposed by Occ. In fact, it holds $L_{\mathcal{K}}(\mathcal{A}) = L_{\mathcal{K}}(\mathcal{B})$.

Converting a formula φ into an ACAT is effective, because converting φ into a graph acceptor is: a radius R and a threshold n can be computed so that $L_{\mathcal{K}}(\varphi)$ is the finite union of $\sim_{r,n}$-equivalence classes, which do not distinguish graphs in which any sphere of radius R appears more than n times or equally often [9]. In turn, the equivalence classes of $\sim_{r,n}$ can be captured by a graph acceptor. $\square$

4 Related Work and Applications

ACA(T)s cover many other models for concurrency, among them CFMs, lossy channel systems, and asynchronous (trace) automata. Let us discuss the former two in more detail, while the relation to the latter is studied thoroughly in [6].

Unlike the general model of an ACAT, a CFM [3] is tailored to recognizing MSCs. It comprises a collection of finite-state machines, one for each agent $i \in Ag$, which communicate with one another via unbounded FIFO channels. The machine of agent i can execute send actions $i!j$ and receive actions $i?j$, $i \neq j$, with the understanding that some message is queued into the channel from agent i to agent j and, respectively, taken out of the (distinct) channel from j to i. To increase the expressiveness, synchronization data might be sent together with the actual message, which are abstracted away in the recognized MSC.

The first logical characterization of CFMs imposed a bound on the channel capacity for any possible computation. In that case, CFMs turned out to be exactly as expressive as MSO logic [10]. Weakening this restriction and considering a fragment of MSCs that requires at least one computation to match a given channel capacity, Genest et al. showed that CFMs still capture precisely MSO logic [8]. Applying the technique that has been generalized in the present paper, CFMs could be shown to be expressively equivalent to EMSO logic over graphs if channels are assumed to be unbounded [2], which is subsumed by Theorem 2.

Not only are CFMs expressively equivalent to ACA(T)s relative to $\mathbb{MSC}(Ag)$, any CFM is precisely an ACA. For example, a transition $(q, i!j, m', q')$ (with synchronization message m') of a CFM component can be written as the collection of ACA transitions $\{(i\theta k, (m, q))\} \rightarrow (i!j, (m', q'))$ where m is a synchronization message, $k \neq i$, and $\theta \in \{!, ?\}$. In particular, a state of the CFM is paired off with a message to obtain a state of the ACA under construction. On the other hand, reading a receive vertex, the only valid ACA transition that may contribute to a run on some MSC is of the form $\{(i\theta k, q_1), (j!i, q_2)\} \rightarrow (i?j, q)$ where the exchange of a message is hidden behind the states. Altogether, we can justifiably state that CFMs are precisely ACA(T)s relative to $\mathbb{MSC}(Ag)$.

If, in a CFM, messages may get lost, we deal with a lossy channel system [1]. Paradoxically, some questions of interest become decidable in this faulty case while, wrt. reliable CFMs, corresponding problems are undecidable [4]. This fact appears less paradox if we consider a lossy channel system to be an ACA relative to $\mathbb{LMSC}(Ag)$ rather than relative to $\mathbb{MSC}(Ag)$. As $\mathbb{LMSC}(Ag)$ is not communication closed, EMSO logic is not covered by ACAs relative to $\mathbb{LMSC}(Ag)$, unless we consider typed lossy channel systems. Reachability for lossy channel systems is decidable [1] and so is emptiness for ACAs relative to $\mathbb{LMSC}(Ag)$.

Theorem 4.

(1) Emptiness for ACAs relative to $\mathbb{LMSC}(Ag)$ is decidable.
(2) Emptiness for ACAs relative to $\mathbb{DAG}(\widetilde{\Sigma})$ is decidable.
(3) Emptiness for ACATs relative to $\mathbb{DAG}(\widetilde{\Sigma})$ is undecidable.
(4) Emptiness for ACAs relative to $\mathbb{MSC}(Ag)$ is undecidable.
(5) Emptiness for ACATs relative to $\mathbb{LMSC}(Ag)$ is undecidable.

Proof. Statement (1) follows from [1] and (2) follows from [11] and the fact that $\mathbb{DAG}(\widetilde{\Sigma})$ is the language of an ACA relative to $\mathbb{DAG}((\{a\})_{a\in\Sigma})$. Part (4) is due to [3] and statements (3) and (5) are reductions from (4). $\square$

References

1. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proceedings of LICS 1993*. IEEE Computer Society Press, 1993.
2. B. Bollig and M. Leucker. Message-Passing Automata are expressively equivalent to EMSO Logic. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, 2004.
3. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
4. G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1), 1996.
5. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
6. M. Droste, P. Gastin, and D. Kuske. Asynchronous cellular automata for pomsets. *Theoretical Computer Science*, 247(1-2), 2000.
7. A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.
8. B. Genest, A. Muscholl, and D. Kuske. A Kleene theorem for a class of communicating automata with effective algorithms. In *Proceedings of DLT 2004*, volume 3340 of *LNCS*. Springer, 2004.
9. W. P. Hanf. Model-theoretic methods in the study of elementary logic. In *The Theory of Models*. North-Holland, Amsterdam, 1965.
10. J. G. Henriksen, M. Mukund, K. Narayan Kumar, and P. S. Thiagarajan. Regular collections of message sequence charts. In *Proceedings of MFCS 2000*, volume 1893 of *LNCS*. Springer, 2000.
11. D. Kuske. Emptiness is decidable for asynchronous cellular machines. In *Proceedings of CONCUR 2000*, 2000.
12. O. Matz, N. Schweikardt, and W. Thomas. The monadic quantifier alternation hierarchy over grids and graphs. *Information and Computation*, 179(2), 2002.
13. A. Potthoff, S. Seibert, and W. Thomas. Nondeterminism versus determinism of finite automata over directed acyclic graphs. *Bulletin of the Belgian Mathematical Society*, 1, 1994.
14. W. Thomas. On Logics, Tilings, and Automata. In *Proceedings of ICALP 1991*, volume 510 of *LNCS*. Springer, 1991.
15. W. Thomas. Elements of an automata theory over partial orders. In *Proceedings of POMIV 1996*, volume 29 of *DIMACS*. AMS, 1996.
16. W. Thomas. Automata theory on trees and partial orders. In *Proceedings of TAPSOFT 1997*, volume 1214 of *LNCS*. Springer, 1997.
17. Wiesław Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. — Informatique Théorique et Applications*, 21, 1987.