Part I

Asynchronous cellular machines

Chapter 2

Σ-dags and asynchronous cellular machines

In this chapter, we define Σ-dags and asynchronous cellular automata, the central topics of the first part of the present work. In addition, this chapter contains several examples that hopefully enable an intuition connected to the notions defined.

We start with the definition of a Σ-dag. These directed acyclic graphs generalise an aspect of dependence graphs known from trace theory: As defined in Section 1.2, a dependence graph is a labeled partially ordered set $(V, \leq, \lambda)$. Let $\longrightarrow\!\!<$ denote the associated covering relation. Then $(V, \longrightarrow\!\!<, \lambda)$, called Hasse-diagram of the dependence graph $(V, \leq, \lambda)$, is an example of a Σ-dag. In particular, $\lambda^{-1}(a)$, the set of a-labeled nodes, is linearly ordered by the reflexive and transitive closure of the edge relation $\longrightarrow\!\!<$, and for any node y and any label a, there is at most one a-labeled node x with $x \longrightarrow\!\!< y$ and at most one a-labeled node z with $y \longrightarrow\!\!< z$. These properties define a Σ-dag:

Definition 2.1.1 Let Σ be an alphabet. A Σ-*dag* is a triple (V, E, λ) where (V, E) is a finite directed acyclic graph and $\lambda : V \to \Sigma$ is a labeling function such that

1. $\lambda^{-1}(a)$ is linearly ordered with respect to $E^\star$ for any $a \in \Sigma$, and
2. for any $(x, y), (x', y') \in E$ with $\lambda(x) = \lambda(x')$, $\lambda(y) = \lambda(y')$, we have $(x, x') \in E^\star$ if and only if $(y, y') \in E^\star$.

By $\mathbb{D}$, we denote the set of all Σ-dags.

As usual, we will identify isomorphic Σ-dags. So let $(V, E, \lambda) \in \mathbb{D}$. Since (V, E) is acyclic, $E^\star$ is a partial order on V. By the first requirement, vertices that carry the same label are comparable with respect to the partial order $E^\star$. In computer science, this property is referred to as "no autoconcurrency". In particular, the width of the partially ordered set $(V, E^\star)$ is bounded by $|\Sigma|$ and

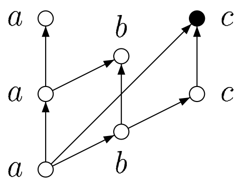

Figure 2.1: An $\{a, b, c\}$-dag

there is a natural covering of a Σ-dag by the $|\Sigma|$ chains $\lambda^{-1}(a)$ for $a \in \Sigma$. Any edge connects two such chains. The second clause ensures in particular that two edges connecting the same chains (in the same direction) cannot "cross". More precisely, let $(x, y), (x', y') \in E$ with $\lambda(x) = \lambda(x')$ and $\lambda(y) = \lambda(y')$, i.e. these two edges connect the same chains in the same direction. Then, by the first requirement, x and x' are comparable with respect to $E^\star$, say $(x, x') \in E^\star$. Then the second requirement forces $(y, y') \in E^\star$. In particular, if $(x, y), (x, y') \in E$ with $\lambda(y) = \lambda(y')$, then $y = y'$ and dually if $(x, y), (x', y) \in E$ with $\lambda(x) = \lambda(x')$, then x and x' are forced to be equal.

Example 2.1.2 1. Let $\Sigma = \{a, b, c\}$. Then the labeled directed acyclic graph depicted in Figure 2.1 is a Σ-dag.

2. Let $(P, \leq)$ be a finite partial order and $\lambda : P \to \Sigma$ be a mapping. Then the triple $(P, \leq, \lambda)$ is a *pomset without autoconcurrency* if, for any $a \in \Sigma$, the set $\lambda^{-1}(a)$ is linearly ordered by $\leq$ (e.g., the left dag in Figure 2.2 is a pomset without autoconcurrency). Note that $(V, E^\star, \lambda)$ is a pomset without autoconcurrency for any Σ-dag (V, E, λ). Conversely, a pomset without autoconcurrency is not a Σ-dag for it may violate the second requirement. Now let $x \longrightarrow\!\!\!< y$ denote that $x < y$ and there is no element properly between x and y (We say that *x is covered by y*). The *Hasse-diagram* $\text{Ha}(P, \leq, \lambda)$ of $(P, \leq, \lambda)$ is the labeled directed acyclic graph $(P, \longrightarrow\!\!\!<, \lambda)$. It is easily checked that this Hasse-diagram is a Σ-dag whenever $(P, \leq, \lambda)$ is a pomset without autoconcurrency (cf. the right dag in Figure 2.2).

Let $t = (V, E, \lambda) \in \mathbb{D}$ be a Σ-dag and $x \in V$. Then the *reading domain* $\text{R}(x)$ of x is the set of all letters a from Σ that satisfy

$$\exists y \in V : \lambda(y) = a \text{ and } (y, x) \in E,$$

i.e. $R(x)$ is the set of labels of those nodes $y \in V$ that are connected with x by an edge (y, x). Informally, these nodes can be seen as the lower neighbors of

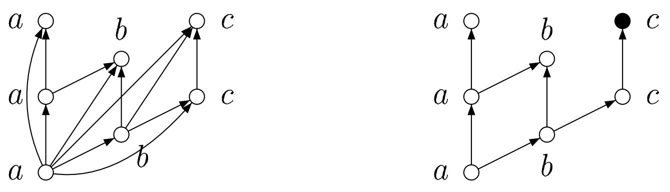

Figure 2.2: A pomset without autoconcurrency and its Hasse-diagram

x in the dag (V, E, λ) (but not necessarily in the partial order $(V, E^\star, \lambda)$). For $a \in \mathrm{R}(x)$ let $\partial_a(x)$ denote the (unique) element y of $\lambda^{-1}(a)$ with $(y, x) \in E$. Thus, $\{\partial_a(x) \mid a \in \mathrm{R}(x)\}$ is the set of lower neighbors of x in the Σ-dag t.

Example 2.1.2 (continued) Let x denote the element of the Σ-dag from Figure 2.1 depicted by a solid circle. Since x is the target of edges whose source is labeled by a and by c, respectively, the reading domain $\mathrm{R}(x)$ is $\{a, c\}$. Differently, the solid circle in the Σ-dag from Figure 2.2 is the target of only one edge whose source is labeled by c. Hence, for this Σ-dag, $\mathrm{R}(x) = \{c\}$.

Next we define asynchronous cellular machines, the model of parallel systems that we are going to investigate. An *asynchronous cellular machine over* Σ or Σ-*ACM* is a tuple

$$\mathcal{A} = ((Q_a, \sqsubseteq_a)_{a\in\Sigma}, (\delta_{a,J})_{a\in\Sigma, J\subseteq\Sigma}, F)$$

where

1. $(Q_a, \sqsubseteq_a)$ is an at most countable, well-quasi ordered set of local states for any $a \in \Sigma$,
2. $\delta_{a,J} : \prod_{b\in J} Q_b \to 2^{Q_a}$ is a nondeterministic transition function for any $a \in \Sigma$, $J \subseteq \Sigma$, and
3. $F \subseteq \bigcup_{J\subseteq\Sigma} \prod_{b\in J} Q_b$ is a finite set of accepting states.

One can think of a Σ-ACM as a Σ-tuple of sequential devices. The device with index a performs the a-labeled events of an execution. Doing so, it reads states from other consitutents of the Σ-ACM. But it changes its own state, only (see below for a formal definition of a run).

A Σ-ACM is *deterministic* if, for any $a \in \Sigma, J \subseteq \Sigma$ and $q_b \in Q_b$ for $b \in J$ the set $\delta_{a,J}((q_b)_{b\in J})$ is a singleton.[1] The set of all local states $\bigcup_{a\in\Sigma} Q_a$ will be denoted by Q.

A Σ-ACM is *monotone* if, for any $a \in \Sigma$, $J \subseteq \Sigma$, $p_b, p'_b \in Q_b$ for $b \in J$ and $q \in Q_a$, we have

$$q \in \delta_{a,J}((p_b)_{b\in J}) \text{ and } p_b \sqsubseteq_b p'_b \text{ for } b \in J \Longrightarrow \exists q' \in \delta_{a,J}((p'_b)_{b\in J}) : q \sqsubseteq_a q'.$$

Intuitively, this means that increasing the input of a transition does not disable the transition and increases its output.

A Σ-ACM is called *asynchronous cellular automaton* over Σ (Σ-*ACA* for short) provided the sets of local states Q_c are finite for $c \in \Sigma$. Usually, for an ACA we will assume the wqos $\sqsubseteq_c$ to be the trivial reflexive relation Δ_{Q_c} on Q_c. Hence, any asynchronous cellular automaton is monotone.

Example 2.1.3 Let Σ be an alphabet. For $a \in \Sigma$ let $Q_a := \mathbb{N}^\Sigma$ be the set of all functions $\Sigma \to \{0, 1, 2, \ldots\}$. The local wqos $\sqsubseteq_a$ are defined by $f \sqsubseteq_a g$ iff $f(b) \le g(b)$ for any $b \in \Sigma$. Next, we define the transition function by

$$\delta_{a,J}((f_c)_{c\in J}) := \begin{cases} \emptyset & \text{if there exist } c, d \in J \text{ with } c \neq d \text{ and } f_c(c) \le f_d(c) \\ \{g\} & \text{otherwise} \end{cases}$$

where the function $g : \Sigma \to \mathbb{N}$ is given by

$$g(b) := \begin{cases} \sup\{f_c(b) \mid c \in J\} & \text{if } a \neq b \\ 1 + \sup\{f_c(b) \mid c \in J\} & \text{if } a = b. \end{cases}$$

Furthermore, F is the set of all tuples $(f_c)_{c\in J}$ for $J \subseteq \Sigma$ with $f_c(b) \in \{0, 1\}$ for all $b \in \Sigma$ and $f_c(c) = 0$ for $c \in J$. The Σ-ACM $\mathcal{A} = ((Q_a, \sqsubseteq_a)_{a\in\Sigma}, (\delta_{a,J})_{a\in\Sigma, J\subseteq\Sigma}, F)$ is not deterministic since in some cases $\delta_{a,J}((f_c)_{c\in J})$ is the empty set. Defining $Q'_a := Q_a \dot\cup \{\bot\}$ with $\bot \sqsubset_a f$ for $f \in Q_a$ and $\delta'_{a,J}((f_c)_{c\in J}) = \{\bot\}$ in all cases where it was undefined so far, we obtain a deterministic Σ-ACM $\mathcal{A}'$. Note that this ACM is not monotone. We will return to this example later and show that $\mathcal{A}$ accepts the set of all Hasse-diagrams of pomsets without autoconcurrency.

Next we define how a Σ-ACM can run on a Σ-dag and when it accepts a Σ-dag. Let $t = (V, E, \lambda)$ be a Σ-dag and $\mathcal{A}$ a Σ-ACM. Let $r : V \to \bigcup_{a\in\Sigma} Q_a$ be a mapping and $x \in V$ be a node of t. Then r *satisfies the run condition of* $\mathcal{A}$ *at* x *(relative to* t*)* if

$$r(x) \in \delta_{\lambda(x),\mathrm{R}(x)}((r\partial_b(x))_{b\in\mathrm{R}(x)}).$$

[1]Note that a deterministic Σ-ACM is "complete" since $\delta_{a,J}((q_b)_{b\in J}) \neq \emptyset$. As usual, this is no proper restriction since "incomplete" Σ-ACMs can be "completed" by the introduction of an additional state.

The mapping r is a *run of $\mathcal{A}$ on t* if it satisfies the run condition for any $x \in V$. Note that, for a run r and $x \in V$, we always have $r(x) \in Q_{\lambda(x)}$ since the image of $\delta_{\lambda(x),\mathrm{R}(x)}$ belongs to $Q_{\lambda(x)}$.

Although the transitions of a Σ-ACM $\mathcal{A}$ are modeled by functions $\delta_{a,J}$, we can think of them as tuples $(q,(p_b)_{b\in J})$ with $q \in \delta_{a,J}((p_b)_{b\in J})$. Such a tuple can be understood as a directed acyclic graph with node set $\{q, p_b \mid b \in J\}$ and edges from p_b to q for $b \in J$. Furthermore, the nodes are labeled by elements of $\Sigma \times Q$ where q gets the label (a,q) and p_b is labeled by (b,p_b). Note that on the other side a run r on a Σ-dag $t = (V,E,\lambda)$ defines a (Σ,Q)-labeled dag by $t' = (V,E,\lambda \times r)$. Then r is a run iff for any $y \in V$, the restriction of t' to y and its lower neighbors is a transition, i.e. if t' can be "tiled" by the transitions. Differently from nondeterministic graph acceptors, considered e.g. in [Tho97b, Tho97a], here we take only into account the lower neighbors and not all neighbors of a node y. The reason for this restriction is that we understand a Σ-dag as a process that continues in time. Having this in mind, it is clear that the state reached by performing a node y can depend only on its history but not on the future.

Now let r be a run on the Σ-dag $t = (V,E,\lambda)$. It is *successful* provided there exists a tuple $(q_a)_{a\in\lambda(V)} \in F$ with

$$r(\max(\lambda^{-1}(a))) \sqsupseteq_a q_a \text{ for all } a \in \lambda(V),$$

i.e. if the "global final state" $(r(\max\{\lambda^{-1}(a)\}))_{a\in\lambda(V)}$ dominates some accepting state in the direct product of the wqos $\sqsubseteq_a$. Let $L(\mathcal{A}) \subseteq \mathbb{D}$ denote the set of all Σ-dags that admit a successful run of $\mathcal{A}$. Let M be a set of Σ-dags and $L \subseteq M$. Then we say that L *can be accepted by a Σ-ACM relative to M* if there is a Σ-ACM $\mathcal{A}$ with $L(\mathcal{A}) \cap M = L$. Sometimes we will omit the term "relative to M" if the set M is clear from the context or if M is the set of all Σ-dags. The word "recognize" is reserved for asynchronous cellular automata, i.e. a set of Σ-dags L is *recognizable relative to M* if there exists a Σ-ACA $\mathcal{A}$ with $L = L(\mathcal{A}) \cap M$.

Example 2.1.3 (continued) Let Σ be an alphabet and let Ha denote the set of Hasse-diagrams of pomsets without autoconcurrency. Then $\mathrm{Ha} \subseteq \mathbb{D}$. Furthermore, let $\mathcal{A}$ denote the Σ-ACM defined above. We show that $L(\mathcal{A}) = \mathrm{Ha}$: For a Hasse-diagram $(P, -\!\!<, \lambda) \in \mathrm{Ha}$ let $r(x)(a)$ be the number of a-labeled elements below x (possibly including x, cf. Figure 2.3 for an example where a tuple (x,y,z) denotes the function $\{(a,x),(b,y),(c,z)\}$). For $x \in P$, the reading domain $\mathrm{R}(x)$ is the set of labels of vertices covered by x. Thus, the vertices $\partial_c(x)$ for $c \in \mathrm{R}(x)$ are mutually incomparable. Hence, for $c \in \mathrm{R}(x)$, the vertex $\partial_c(x)$ dominates the largest number of c-labeled vertices among $\{\partial_d(x) \mid d \in \mathrm{R}(x)\}$. Hence $r(\partial_c(x))(c) > r(\partial_d(x))(c)$ for $d \in \mathrm{R}(x) \setminus \{c\}$, i.e. r is a run of $\mathcal{A}$ on $(P, -\!\!<, \lambda)$. Since any tuple $(g_c)_{c\in J}$ dominates some state from F, this run r is accepting, i.e. $\mathrm{Ha} \subseteq L(\mathcal{A})$. Conversely, let r be a successful run of $\mathcal{A}$ on the Σ-dag

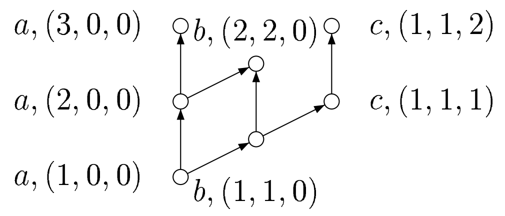

Figure 2.3: A run of $\mathcal{A}$

(V, E, λ). By $\sqsubseteq$, we denote the reflexive and transitive closure of the edge relation E. Then, for any $x \in V$, $c \in \mathrm{R}(x)$ and $a \in \Sigma$ we have $r(\partial_c(x))(a) \leq r(x)(a)$, i.e. $h_a : (V, E^\star) \to \mathbb{N}$ defined by $h_a(x) := r(x)(a)$ is monotone with respect to the partial order $(V, E^\star)$. Furthermore, by the definition of $\delta_{a,J}$, for any $x \in V$ and $c, d \in \mathrm{R}(x)$ with $c \neq d$ we have $h_c(\partial_c(x)) > h_c(\partial_d(x))$. Hence $\partial_c(x) \not\sqsubseteq \partial_d(x)$. Since we can similarly infer $\partial_d(x) \not\sqsubseteq \partial_c(x)$, the elements $\partial_c(x)$ and $\partial_d(x)$ are incomparable. Hence (V, E, λ) is a Hasse-diagram. Thus, the set of Hasse-diagrams can be accepted by a Σ-ACM with infinitely many states. Recall that the ACM $\mathcal{A}$ is not monotone. It is not known whether there exists a monotone ACM $\mathcal{A}'$ equivalent to $\mathcal{A}$, in particular, I do not know whether finitely many states suffice. On the other hand, Lemma 4.1.3 below will show that the set of *not*-Hasse-diagrams can be accepted by a Σ-ACA, i.e. by a Σ-ACM with only finitely many states.

Example 2.1.4 Let L be the set of all Σ-dags t satisfying

the number of d-labeled vertices of t is even for any $d \in \Sigma$.

This set can be accepted by a Σ-ACM that differs from the ACM $\mathcal{A}$ from Example 2.1.3 only in the wqos $\sqsubseteq_a$: Here, we define $f \sqsubseteq_a g$ iff $f(b) \leq g(b)$ for $b \in \Sigma$ and $f(a) \equiv g(a) \mod 2$. Then a tuple $(f_c)_{c \in \Sigma}$ dominates some accepting state iff $f_c(c)$ is even for all $c \in J$. Consider the run of $\mathcal{A}$ on the Σ-dag in Figure 2.3: The maximal a-labeled vertex carries a state f_a with $3 = f_a(a)$. Furthermore, let f_b and f_c denote the state at the maximal b-labeled and c-labeled vertex, respectively. Then the tuple (f_a, f_b, f_c) does not dominate (in the wqo $\sqsubseteq_a \times \sqsubseteq_b \times \sqsubseteq_c$) any state from F, i.e. the run r is not successful.

We finish this chapter with some examples of the expressive power of monotone ACMs. In the first of these examples, we consider ACMs that run on words over Σ. To do this, we identify a word over Σ with the Hasse-diagram of a linearly ordered Σ-labeled poset. In this sense, we can show that the "word-language"

$\{a^m b^n \mid 1 \leq n < m\}$ can be accepted by a monotone ACM. This in particular implies that monotone ACMs are more powerful than finite automata since ACMs can have infinite sets of states. In addition, we will show that the set $\{b^n a^m \mid 1 \leq n < m\}$ cannot be accepted by a Σ-ACM. Thus, the set of languages acceptable by a monotone ACM is not closed under reversal. This might be surprising at first glance, but it is not really so since the notion of well quasi ordering as well as that of monotonicity are not symmetric.

Example 2.1.5 Let $\Sigma = \{a, b\}$ and $L = \{a^m b^n \mid 1 \leq n < m\} \subseteq \Sigma^+$. We consider the words in Σ^+ as Hasse-diagrams of linearly ordered sets so that $L \subseteq \mathbb{D}$. Let $\mathcal{A}$ denote the following Σ-ACM:
$Q_a = Q_b = \mathbb{N}$ with the usual linear order which is a wqo,

$$\delta_{a,J}((q_c)_{c\in J}) = \begin{cases} \emptyset & \text{if } b \in J \\ \{1\} & \text{if } J = \emptyset \\ \{q_a + 1\} & \text{otherwise, i.e. if } J = \{a\}, \text{ and} \end{cases}$$

$$\delta_{b,J}((q_c)_{c\in J}) = \begin{cases} \emptyset & \text{if } J = \{a, b\} \text{ or } J = \emptyset \\ \{\max(0, q_a - 1)\} & \text{if } J = \{a\} \\ \{\max(0, q_b - 1)\} & \text{otherwise, i.e. if } J = \{b\}. \end{cases}$$

The state $(1, 1)$ is the only accepting state from F. We show that the only Hasse-diagrams of linearly ordered sets that are accepted by $\mathcal{A}$ are those from the set L: So let $(V, E, \lambda) \in L$. It is of the form

$$\begin{tikzcd} a_1 \arrow[r, dash] & a_2 \arrow[r, dash] & \ldots \arrow[r, dash] & a_m \arrow[r, dash] & b_1 \arrow[r, dash] & b_2 \arrow[r, dash] & \ldots \arrow[r, dash] & b_n \end{tikzcd}$$

with $1 \leq n < m$, $\lambda(a_i) = a$ and $\lambda(b_i) = b$ for all suitable i. Then the mapping $r : V \to \mathbb{N}$ with $r(a_i) = i$ and $r(b_i) = m - i$ is a run of $\mathcal{A}$ on (V, E, λ). It is successful since the final global state $(q_c)_{c\in\Sigma}$ equals $(m, m - n)$ and $m - n \geq 1$. If, on the contrary, (V, E, λ) is the Hasse-diagram of a linear order, but not from L, then
either it contains some a-labeled vertex that covers a b-labeled one,
or it contains some b-labeled vertex which is not the target of any edge,
or it is of the form $a^m b^n$ with $m \leq n$.

In the first case, the ACA $\mathcal{A}$ does not have any run on (V, E, λ) due to $\delta_{a,J}((q_c)_{c\in J}) = \emptyset$ whenever $b \in J$. Similarly in the second case, since $\delta_{b,\emptyset} = \emptyset$. In the third case, there is a run of $\mathcal{A}$ on (V, E, λ), but the final global state is of the form $(m, 0)$ and therefore not successful.

Now let $L' = \{b^n a^m \mid 1 \leq n < m\}$ denote the set of reversed words from L. We show that there is no monotone Σ-ACM $\mathcal{A}'$ that accepts the Hasse-diagrams that correspond to words in L' relative to the Hasse-diagrams of linear orders: By contradiction, assume there is such a Σ-ACM $\mathcal{A}'$. For $n > 0$ let t_n denote the Hasse-diagram corresponding to $b^n a^{n+1}$. Since these words belong to L', there exists a successful run r_n of $\mathcal{A}'$ on t_n for any $n > 0$. Let q_n denote the state that is associated by r_n to the last b-labeled vertex in t_n (i.e. $q_n \in Q_b$ is the state of $\mathcal{A}'$ that is reached after performing the b-prefix of $b^n a^{n+1}$). Since $\sqsubseteq_b$ is a wqo, there are $n < m$ with $q_n \sqsubseteq_b q_m$. Now consider the Hasse-diagram t associated to the word $b^m a^{n+1}$. Since $n < m$, this word does not belong to L'. Nevertheless, since $\mathcal{A}'$ is monotone, there is a successful run on $t = (V, E, \lambda)$: This Σ-dag has the form:

$$\begin{tikzcd} b_1 \arrow[r, dash] & b_2 \arrow[r, dash] & \ldots \arrow[r, dash] & b_m \arrow[r, dash] & a_1 \arrow[r, dash] & a_2 \arrow[r, dash] & \ldots \arrow[r, dash] & a_n \end{tikzcd}$$

The mapping $r_m \restriction \{b_1, b_2, \ldots, b_m\}$ satisfies the run conditions for b_i relative to t since r_m is a run of $\mathcal{A}'$ on t_m. Furthermore, $r_m(b_m) = q_m \sqsupseteq_b q_n$. Since $r_n(a_1) \in \delta_{a,\{b\}}(q_n)$ and since $\mathcal{A}'$ is monotone, there exists $r(a_1) \in \delta_{a,\{b\}}(q_m)$ with $r_n(a_1) \sqsubseteq_a r(a_1)$. By induction, we obtain states $r(a_i) \sqsupseteq_a r_n(a_i)$ such that $r_m \restriction \{b_1, b_2, \ldots, b_m\} \cup r$ is a run of $\mathcal{A}'$ on t. Since $r(a_{n+1}) \sqsupseteq_a r_n(a_{n+1})$ and $r_m(b_m) \sqsupseteq_b r_n(b_n)$, the final global state $(r_m(b_m), r(a_{n+1}))$ of this run dominates that of r_n which equals $(r_n(b_n), r_n(a_{n+1}))$. But r_n was successful, hence so is this new run, i.e. t is accepted by $\mathcal{A}'$.

Note that the language $\{b^n a^m \mid 1 \leq n < m\}$ cannot be accepted by a finite sequential automaton. Hence, it is not monadically axiomatizable. The last example in this chapter gives an elementarily axiomatizable set of Σ-dags that cannot be accepted by a monotone ACM:

Example 2.1.6 Let $\Sigma = \{a, b\}$ and let φ denote the first-order sentence

$$\forall x \exists y((\lambda(x) = a) \rightarrow ((\lambda(y) = b) \wedge (x, y) \in E)).$$

Note that a Σ-dag satisfies φ iff every a-labeled element is the source of an edge that leads to a b-labeled vertex. Furthermore, let L be the set of all Σ-dags that satisfy φ. We show that L cannot be accepted by a monotone Σ-ACM:

By contradiction, we assume $\mathcal{A}$ to be a Σ-ACA such that $L(\mathcal{A}) = L$. For $n \in \mathbb{N}$ consider the Σ-dag $t_n = (V_n, E_n, \lambda_n)$ defined as follows: The set V_n equals $\{a_i, b_i \mid i = 1, 2, \ldots, n\}$ with the edge relation

$$\{(a_i, a_{i+1}) \mid 1 \leq i < n\} \cup \{(b_i, b_{i+1}) \mid 1 \leq i < n\} \cup \{(a_i, b_i) \mid 1 \leq i \leq n\}$$

and the labeling $\lambda_n(a_i) = a$ and $\lambda_n(b_i) = b$ for $1 \leq i \leq n$ (cf. the first Σ-dag in Figure 2.4 for the case $n = 8$).

Recall that φ states that every element labeled by a is the source of an edge leading to an element labeled by b. Hence $t_n \in L$. By the assumption that $\mathcal{A}$ accepts those Σ-dags that satisfy φ, there exists a successful run r_n of $\mathcal{A}$ on t_n for all $n \in \mathbb{N}$. Let w_n denote the word $r_n(a_1) r_n(a_2) r_n(a_3) \ldots r_n(a_n) \in Q_a^\star$. By Higman's Theorem [Hig52] (page 2), there exist $m < n$ such that w_m is dominated by a subword of w_n that contains the last position, i.e. such that there exist $1 \leq j_1 < j_2 < \cdots < j_m = n$ with $r_m(a_i) \sqsubseteq_a r_n(a_{j_i})$. Now consider the

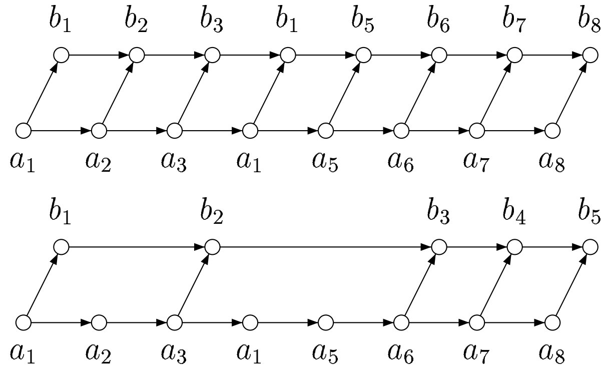
b_1 b_2 b_3 b_1 b_5 b_6 b_7 b_8
a_1 a_2 a_3 a_1 a_5 a_6 a_7 a_8
b_1 b_2 b_3 b_4 b_5
a_1 a_2 a_3 a_1 a_5 a_6 a_7 a_8

Figure 2.4: cf. Example 2.1.6.

Σ-dag $t = (V, E, \lambda)$ defined by $V = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$, $\lambda(a_i) = a$ and $\lambda(b_i) = b$ for all suitable i and the edge relation

$$\{(a_i, a_{i+1}) \mid 1 \leq i < n\} \cup \{(b_i, b_{i+1}) \mid 1 \leq i < m\} \cup \{(a_{j_i}, b_i) \mid 1 \leq i \leq m\}$$

(see the second Σ-dag in Figure 2.4 with $m = 5, n = 8, j_1 = 1, j_2 = 3, j_3 = 6, j_4 = 7$, and $j_5 = 8$). Then $t \notin L$. On the other hand we construct a successful run r of $\mathcal{A}$ on t as follows: Let $r(a_i) = r_n(a_i)$ for $1 \leq i \leq n$. Then r satisfies the run condition at a_i since r_n is a run. Recall that $r_m(a_i) \sqsubseteq_a r_n(a_{j_i}) = r(a_{j_i})$. Since $\mathcal{A}$ is monotone, there exists $r(b_1) \in \delta_{b,\{a\}}(r(a_{j_1}))$ with $r_n(b_1) \sqsubseteq_b r(b_1)$. Inductively, we find $r(b_i) \in \delta_{b,\{a,b\}}(r(a_{j_i}), r(b_{i-1}))$ with $r(b_i) \sqsupseteq_b r_m(b_i)$. At the end, $r(b_m) \sqsupseteq_b r_m(b_m)$. Thus, the final global state $(r(a_n), r(b_m))$ dominates $(r_m(a_m), r_m(b_m))$. Since r_m was successful, so is r, i.e. $t \in L(\mathcal{A})$, a contradiction.

On the other hand, the set $\mathbb{D} \setminus L$ can easily be accepted by a Σ-ACA, i.e. with only finitely many states. A Σ-dag does not belong to L iff it has an a-labeled vertex that is not the source of any edge connecting it to a b-labeled vertex, i.e. the state associated to this vertex by a possible run is not seen by any b-labeled vertex. Thus, the idea of a Σ-ACA that accepts the complement of L is to nondeterministically pick at least one a-labeled vertex and mark it by a distinguished state. The b-transitions just have to check that they do not read this distinguished state.