State Machines (In Modular Petri Nets)

Sophie Wallner

Universität Rostock

A *modular Petri net* is a composition of individual Petri net systems that show both - individual internal behavior and synchronized interface behavior [4] [1] [3]. The main benefit of modular Petri nets is the parallel interleaved computation of the component's state spaces [3] to counteract the state explosion problem. As each component's state space is calculated individually, we can exploit the structure of the component for verification. By taking advantage of the heterogeneity of the components we can accelerate the state space generation and facilitate the verification process. In this work, we take a closer look at the Petri net structure *state machines* and show how to exploit its characteristics in the context of a modular Petri net.

Definition 1 (State Machine). *A* state machine *is a Petri net system* $[N, m_0]$*, where every transition has exactly one pre- and one post place, i.e.* $\forall t \in T : |\bullet t| = |\{p \mid (p,t) \in F\}| = 1$ *and* $|t\bullet| = |\{p \mid (t,p) \in F\}| = 1$.

A State Machine $[N, m_0]$ decays into SCCs $\{K_1, \ldots, K_n\}$ where $K_i \subseteq P$ for $i \in \{1, \ldots, n\}$ induce state machines themselves. SCCs only have ingoing and outgoing arcs from resp. to places: For SCC K_i with $i \in \{1, \ldots, n\}$, we accumulate those transition as $\bullet K_i = \{t \mid (t,p) \in F_j, p \in K_i\}$ and $K_i\bullet = \{t \mid (p,t) \in F, p \in K_i\}$. SCC K_i is a *predecessor* of SCC $K_{i'}$ if $(K_i\bullet) \cap (\bullet K_{i'}) \neq \emptyset$ for $i, i' \in \{1, \ldots, n\}$. With this we can introduce a *partial order* $\leq$ such that $K_i \leq K_{i'}$ if K_i is a predecessor of $K_{i'}$, and $K_i \leq K_{i''}$ if $K_{i'} \leq K_{i''}$ and K_i is a predecessor of $K_{i'}$. For SCC K_i with $i \in \{1, \ldots, n\}$, the *backward cone* R_i is the set of itself and all of its transitive predecessors, i.e. $R_i = \{K_{i'} \mid K_{i'} \leq K_i\}$. Backward cones of SCCs can be interleaved.

For a state machine, we can build the *reachability graph* as usual; let $R = [V, E]$ be the reachability graph of $[N, m_0]$ such that $V = RS(\{m_0\})$ and $(m, t, m') \in E$, iff $m \xrightarrow{t} m'$. Every marking m can be abstracted to an *SCC-marking* μ that contains the number of tokens in each SCC of $[N, m_0]$.

Definition 2 (SCC-marking). *For marking* m*, we define the according* SCC-marking $\mu : m \times \{K_1, \ldots, K_n\} \to \mathbb{N}$ *such that* $\mu(m, K_i) = \sum_{p \in K_i} m(p)$ *for* $i \in \{1, \ldots, n\}$.

The following statements follow directly from the state machine structure:

1. The number of tokens in every reachable marking is constant, i.e. $\forall m \in V : \sum_{p \in P} m(p) = const$. No Token cannot be created or vanished because all weights are 1 and every transition has a constant in-out-ratio.

2. Let m be a reachable marking and μ the according SCC-marking. Then, any marking m' with $\mu(m', K_i) = \mu(m, K_i)$ for all $i \in \{1, \ldots, n\}$ is reachable from m. Within each K_i, every distribution of tokens over places is possible because of the strong connectedness. So, the reachability of m' can be reduced to the reachability its corresponding SCC-marking μ.
3. Let $m \in V$ be a reachable marking and μ the according SCC-marking. Then, Marking m' with $\mu(m', K_i) = \sum_{K_l \in R_i} \mu(m, K_l)$ for all $i \in \{1, \ldots, n\}$ is reachable from m.

In General, reachability of markings depends on moving the right number of tokens to the right SCCs without using tokens multiple times. The reachability of marking m' from marking m can be expressed as a linear system of equations (ILP) $\Pi(m, m')$ that results in a solution vector $\boldsymbol{x} \in \mathbb{N}^{n \cdot n}$.

Theorem 1 (Linear System of Equations For Reachability Analysis). *Let m be a reachable marking of $[N, m_0]$. Marking m' is* reachable *from m, iff ILP $\Pi(m, m')$ has an integer solution $\boldsymbol{x} > 0$. The ILP has the form*

$$\mu(m, K_i) + \sum_{l=1}^{n} x_{li} - \sum_{l=1}^{n} x_{il} = \mu(m', K_i) \qquad \forall i \in \{1, \ldots, n\} \quad (1)$$

$$x_{li} = 0 \qquad \forall l, i \in \{1, \ldots, n\} \text{ if } K_l \notin R_i \quad (2)$$

We can use this for the verification of properties in state machines. *State predicates* serve as a base for the verification.

Definition 3 (State Predicate, Visible Places, Coefficient-Mapping). *Let $[N, m_0]$ be a Petri net system. An* atomic state predicate *φ is a comparison of formal sum to an integer $a_1 m(p_1) \cdot \ldots \cdot a_k m(p_k) \leq a$ where $a_1, \ldots, a_k, a \in \mathbb{Z}$ and $k \leq |P|$. An atomic state predicate is satisfied in a marking, if the inequality is true. We call the occurring places $p_1, \ldots, p_k$ in φ* visible*, defining a set $vis(\varphi) \subseteq P$. Given an atomic state predicate φ, we can access the coefficient for visible place p with the* coefficient mapping *$\alpha(p, \varphi) = a$. A* state predicate *is a conjunction $\Phi = \varphi^1 \wedge \ldots \wedge \varphi^g$, where φ^h is an atomic state predicate for $h \in \{1, \ldots, g\}$. A state predicate is satisfied in a marking, if every atomic state predicate is true. We set $vis(\Phi) = \bigcup_{h \in \{1, \ldots, g\}} vis(\varphi^h)$.*

We want to reduce the verification of a state predicate as well to solving an ILP. Therefore, we need to introduce new variables: For each visible place $p \in vis(\Phi)$, we introduce variable p that is part of the solution of the ILP. A solution assigning value $v \in \mathbb{N}$ to p corresponds to a marking m where $m(p) = v$. We build the ILP $\Pi(m_0, \Phi)$ with reference to Theorem. 1 with little adaptions: For every K_i with $i \in \{1, \ldots, n\}$, $\mu(m_0, K_i) + \sum_{l=1}^{n} x_{li} - \sum_{l=1}^{n} x_{il}$ relaxes to the maximum bound for tokens on the visible places in K_i, and every atomic state predicate φ^h needs to be satisfied in one marking.

Theorem 2 (Satisfiability of State Predicates). *Let $\Phi = \varphi^1 \wedge \ldots \wedge \varphi^g$ be a state predicate where φ^h is an atomic state predicate for $h \in \{1, \ldots, g\}$. State*

Predicate Φ is satisfiable *in $[N, m_0]$, iff the following linear system of equations $\Pi(m_0, \Phi)$ has an integer solution $\boldsymbol{xp} > 0$:*

$$x_{li} = 0 \, \forall l, i \in \{1, \ldots, n\} \text{ if } K_l \notin R_i \tag{3}$$

$$\sum_{p \in K_i \cap vis(\Phi)} p \leq d_i + \sum_{l=1}^{n} x_{li} - \sum_{l=1}^{n_j} x_{il} \, \forall i \in \{1, \ldots, n\} \tag{4}$$

$$\sum_{i=1}^{n} \sum_{p \in K_i \cap vis(\varphi^h)} \alpha(p, \varphi^h) p \leq a^h \, \forall h \in \{1, \ldots, g\} \tag{5}$$

Furthermore, we can use the theory on state machines for verification of formulas of the temporal logic $CTL_1 \subset CTL$ only consists of state predicates that may be extended with **EF** to express the possibility of satisfaction or with **AG** to express the necessity of satisfaction. This is in style of the branching time logic L_1 from [2] for 1-safe Petri nets; an extension of propositional logic with a *possibility* operator $\Diamond$ and the *necessity* operator $\Box$. As their theoretical foundation is the same, L_1 and CTL_1 provide the same seven equivalence classes of combinations of the Operators:[], [**EF**], [**AG**], [**EFAG**], [**AGEF**], [**EFAGEF**], [**AGEFAG**]. The goal is to reduce verification of properties of those classes to solving an ILP as well. For example, with Theorem 2 we can verify properties of class [**EF**] directly; State Machine N satisfies **EF**Φ if and only if $\Pi(m_0, \Phi)$ has a nontrivial integer solution. Our goal is to evaluate the verification of the other classes and find a systematic approach that we can embed in our verification portfolio [5].

Finally, we want to adapt this knowledge on state machines to modular Petri nets where some components are state machines. It is more likely that individual components are a state machine than that the entire system is. It would also be possible to modularize the Petri net system such that state machine instances are created. State machines lead to twofold advantages here: In the building process of the state spaces and in the verification of properties.

References

1. Christensen, S., Petrucci, L.: Modular Analysis of Petri Nets. The Computer Journal **43**(3), 224–242 (Jan 2000)
2. Esparza, J., Heljanko, K.: Unfoldings: a partial-order approach to model checking. Springer Science & Business Media (2008)
3. Gaede, J., Wallner, S., Wolf, K.: Modular state spaces-a new perspective. In: International Conference on Applications and Theory of Petri Nets and Concurrency. pp. 312–332. Springer (2024)
4. Latvala, T., Mäkelä, M.: LTL Model Checking for Modular Petri Nets. In: Cortadella, J., Reisig, W. (eds.) Applications and Theory of Petri Nets 2004. pp. 298–311. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004)
5. Schmidt, K.: Lola a low level analyser. In: Application and Theory of Petri Nets 2000: 21st International Conference, ICATPN 2000 Aarhus, Denmark, June 26–30, 2000 Proceedings 21. pp. 465–474. Springer (2000)