Towards Synthesis of Petri Nets from General Partial Languages

Robert Lorenz

Lehrprofessur für Informatik
Universität Augsburg, Germany
e-mail: `robert.lorenz@informatik.uni-augsburg.de`

Abstract. In this paper we investigate synthesis of place/transition Petri nets from three different finite representations of infinite partial languages, generalizing previous results.

1 Introduction

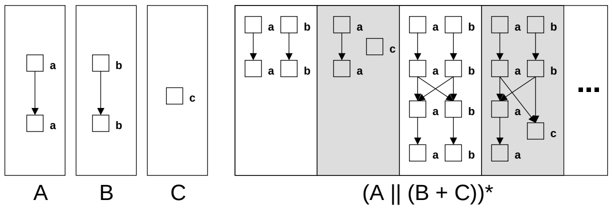


Fig. 1. Partial language given by a term.

In the last two years we generalized the theory of regions for the synthesis of Petri nets from sequential languages and step languages to so called partial languages [LJ06]. A partial language specifies the behaviour of a concurrent system through a possibly infinite set of labeled partial orders (LPOs). Each LPO specifies a run of the system given by a partial order between events labeled by action names. Unordered events are interpreted to be concurrent. The left side of Figure 1 shows three different LPOs, the right side shows a partial language. Through the theory of regions it is possible to compute from a given partial language a Petri net having all specified LPOs as partially ordered runs and having minimal additional behaviour.

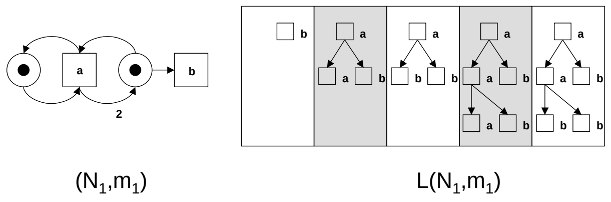


Fig. 2. Partial language without term-based representation.

In this paper we consider classical place/transition Petri nets (p/t-nets). In [LBDM07] we developed an effective synthesis algorithm based on the theory of regions from finite partial languages. In [LBDM08] we generalized this result to such infinite partial languages having a finite term based representation using operators for iteration ($*$), sequential composition (;), alternative composition ($+$) and parallel composition ($\|$). Figure 1 shows some of the LPOs of the infinite partial language given by the term $(A \parallel (B + C))^*$ composing elementary LPOs A, B, C.[1]

Unfortunately only a small class of infinite partial languages can be represented in such a term based form. The Figures 2 and 3 show examples of infinite partial languages which can not be given by a term as above. The main reason for that is, that by the iteration operator it is not possible to append events only to a part of an LPO, but only to the whole LPO. In both cases a p/t-net having the given partial language as its set of (partially ordered) runs is shown.

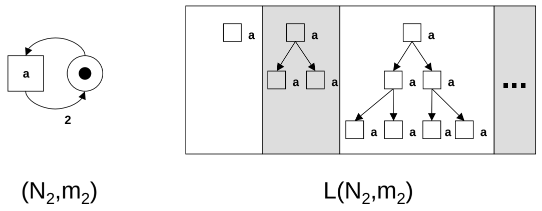


Fig. 3. Partial language without term-based representation.

In this paper we propose three different more general finite representations of infinite partial languages. Each of these representations allows to iteratively append events to parts of LPOs. Therefore, it is possible to represent the finite complete prefix of the branching process of bounded p/t-nets, i.e. we claim that by each of these finite representations the language of (partially ordered runs) of arbitrary bounded p/t-nets can be specified.

Due to lack of space we mostly present the ideas lying behind these finite representations only in an informal way through examples. Finally, very briefly, we suggest how regions could be defined for each of the finite representations.

2 Finite Representations

In this section we introduce three different finite representations of infinite partial languages.

By $\mathbb{N}$ we denote the *nonnegative integers*. $\mathbb{N}^+$ denotes the positive integers. Given a finite set A, the symbol $|A|$ denotes the *cardinality* of A. The set of all *multi-sets* over a set A is the set $\mathbb{N}^A$ of all functions $f : A \to \mathbb{N}$. Given a binary relation $R \subseteq A \times A$, we write aRb to denote $(a, b) \in R$. A *directed graph* is a pair $(V, \to)$, where V is a finite *set of nodes* and $\to \subseteq V \times V$ is called the *set of arcs*. A *partial order* is a directed graph $\mathrm{po} = (V, <)$, where $< \subseteq V \times V$ is irreflexive and transitive.

Definition 1 (Labeled partial order). *A* labeled partial order *(LPO) is a triple* $\mathrm{lpo} = (V, <, l)$*, where* $(V, <)$ *is a partial order and* $l : V \to T$ *is a* labeling function *with* set of labels T.

In our context, a node v of an LPO $(V, <, l)$ is called *event*, representing an occurrence of $l(v)$. Two nodes $v, v' \in V$ are called *independent* if $v \not< v'$ and $v' \not< v$. Notice that by this definition, independence is reflexive. By $\mathrm{co} \subseteq V \times V$ we denote the set of all pairs of independent nodes of V. A *co-set* is a subset $C \subseteq V$ satisfying

[1] Note that for a clearer presentation no transitive edges of the LPOs are drawn.

$\forall x, y \in C : x \operatorname{co} y$. A *cut* is a maximal co-set (w.r.t. set inclusion). For a co-set C of a partial order $(V, <)$ and a node $v \in V \setminus C$ we write $v < C$, if $v < s$ for an element $s \in C$, and $v \operatorname{co} C$, if $v \operatorname{co} s$ for all elements $s \in C$. A partial order $(V', <')$ is a *prefix* of a partial order $(V, <)$ if $V' \subseteq V$, $<' = < |_{V' \times V'}$ and $(v' \in V' \wedge v < v') \Longrightarrow (v \in V')$. Given two partial orders $\mathrm{po}_1 = (V, <_1)$ and $\mathrm{po}_2 = (V, <_2)$, we say that po_2 *is a sequentialization of* po_1 if $<_1 \subseteq <_2$. We use the notations defined for partial orders also for LPOs. If T is the set of labels of $\mathrm{lpo} = (V, <, l)$ then for a set $V' \subseteq V$, we define the multi-set $|V'|_l \subseteq \mathbb{N}^T$ by $|V'|_l(t) = |\{v \in V' \mid l(v) = t\}|$. We consider LPOs only up to isomorphism

Definition 2 (Partial language). *Let T be a set. A set $\mathcal{L}$ of LPOs $\mathrm{lpo} = (V, <, l)$ with $l(V) \subseteq T$ and $\bigcup_{(V,<,l)\in\mathcal{L}} l(V) = T$ is called* partial language over T.

A *net* is a triple (P, T, F), where P is a (possibly infinite) set of *places*, T is a finite set of *transitions* satisfying $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*.

Definition 3 (Place/transition net). *A* place/transition-net *(*p/t-net*)* N *is a quadruple* (P, T, F, W)*, where* (P, T, F) *is a net, and* $W : F \to \mathbb{N}^+$ *is a* weight function.

We extend the weight function W to pairs of net elements $(x, y) \in (P \times T) \cup (T \times P)$ with $(x, y) \notin F$ by $W(x, y) = 0$. A *marking* of a net $N = (P, T, F, W)$ is a function $m : P \to \mathbb{N}$, i.e. a multi-set over P. A *marked p/t-net* is a pair (N, m_0), where N is a p/t-net, and m_0 is a marking of N, called *initial marking*. The occurrence rule of p/t-nets is defined as usual. The non-sequential semantics of a p/t-net can be given by *enabled LPOs*, also called *runs*. An LPO is enabled in a net if the events of the LPO can occur in the net respecting the concurrency relation of the LPO.

Definition 4 (Enabledness). *Let* (N, m_0) *be a marked p/t-net,* $N = (P, T, F, W)$*. An LPO* $\mathrm{lpo} = (V, <, l)$ *with* $l : V \to T$ *is called* enabled w.r.t. (N, m_0) *if for every cut* C *of* lpo *and every* $p \in P$ *there holds* $m_0(p) + \sum_{v \in V \wedge v < C}(W(l(v), p) - W(p, l(v))) \geq \sum_{v \in C} W(p, l(v))$*. Its* occurrence *leads to the marking* m' *given by* $m'(p) = m_0(p) + \sum_{v \in V}(W(l(v), p) - W(p, l(v)))$ *for each* $p \in P$*.*

The set of of LPOs enabled w.r.t. a given marked p/t-net (N, m_0) *is denoted by* $\mathcal{L}(N, m_0)$*.* $\mathcal{L}(N, m_0)$ *is called the* partial language of runs *of* (N, m_0)*.*

An alternative characterization of enabled LPOs is through so called process nets. A process net is an acyclic net without conflicts which “unfolds” a p/t-net by representing tokens from some marking of the p/t-net through places (called conditions) and transition occurrences through transitions (called events). Since in a process net the flow relation has no cycles and thus defines a partial order among conditions and events. Omitting the conditions and keeping this partial order between the events yields an enabled LPO, called run underlying the process net. The other way round, each enabled LPO sequentializes the run underlying some process net.

The set of all (alternative) process nets of a p/t-net can be represented by the (possibly infinite) branching process which is an acyclic net including conflicts. In the case the p/t-net is bounded, there is a finite prefix of the branching process (called complete finite prefix) which represents all reachable markings. Roughly speaking, it is determined through cutting the branching process if a marking is repeated. Omitting the conditions and keeping the partial order and conflict relation between the events yields a so called prime event structure (underlying the finite complete prefix) which represents a set if runs underlying process nets.

Note that the partial language of runs of a p/t-net is always prefix- and sequentialization-closed. In examples and Figures we often do not draw all prefixes and sequentializations but assume that they are present.

2.1 Identification of states

The finite complete prefix (resp. its underlying prime event structure) of a bounded p/t-net can be represented on the level of languages by a finite set of LPOs. Of course, from this finite set the complete non-sequential behavior can only be re-constructed, if one keeps the information, at which points the branching process was cut w.r.t. which repeated marking. This can be done by remembering, which prefixes of which LPO lead to the same marking. That means, a possibility for specifying the non-sequential behavior of bounded p/t-nets is through a finite set of LPOs together with some equivalence relation on prefixes of these LPOs.

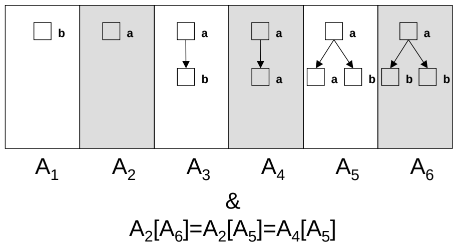


Fig. 4. Set of LPOs with identification of states representing $L(N_1, m_1)$.

If two prefixes are equivalent, this means that all events occurring after the one prefix also can occur after the second prefix and vice versa. Infinite behavior is specified for example if a prefix is prefix of an equivalent prefix. Figure 4 shows, how by this method the language $L(N_1, m_1)$ from Figure 2 can be given. The equation $A_2[A_6] = A_2[A_5] = A_4[A_5]$ means that after occurrence of A_2 in A_6 the same marking is reached as after occurrence of A_2 in A_5 or after occurrence of A_4 in A_5. Therefore, after the occurrence of A_4, the same events as after A_2 in A_5 or A_6 can occur and so on. Also $L(N_2, m_2)$ from Figure 3 can be represented this way (see Figure 5). This means, that through identifying states also the non-sequential behavior of unbounded nets can be specified (at least in some cases).

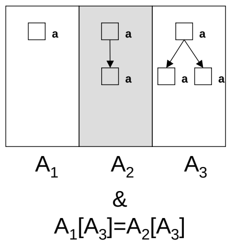
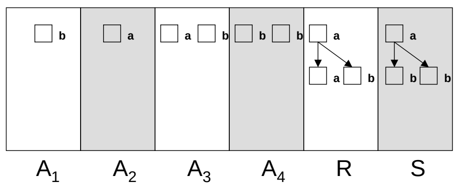


Fig. 5. Set of LPOs with identification of states representing $L(N_2, m_2)$.

2.2 Partial Iteration

In [LBDM08] we introduced a term-based representation of infinite partial language. These terms, called *composed runs*, are build through composing inductively (elementary) LPOs from some given finite set of LPOs $\mathcal{A}$. LPOs can be composed sequentially (;), alternatively ($+$) and parallel ($\|$) and can be iterated ($*$). That means each LPO $A \in \mathcal{A}$ is a composed run and if α, β are composed runs, then also $\alpha;\beta$, $\alpha + \beta$, $\alpha \parallel \beta$ and α^* are composed runs. Each composed run represents a set of LPOs, where an elementary LPO A represents the one-LPO set $L(A) = \{A\}$. The composed run $\alpha;\beta$ represents the set $L(\alpha;\beta) = \{A;B \mid A \in \alpha,\, B \in \beta\}$, $\alpha + \beta$ the set $L(\alpha+\beta) = L(\alpha) \cup L(\beta)$, $\alpha \parallel \beta$ the set $L(\alpha \parallel \beta) = \{A \parallel B \mid A \in \alpha,\, B \in \beta\}$ and α^* the set $L(\alpha^*) = \{A_1; ...; A_n \mid A_i \in \alpha\}$. On the level of LPOs $A;B$ means that each event in A precedes each event in B and $A \parallel B$ means that there is no order between events in A and in B.

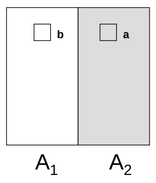
Such a representation of partial languages by composed runs is quite restrictive as shown in the introduction, because through sequential composition and iteration it is not possible to append an LPO only to parts of some previous LPO. We therefore introduce here the possibility to iterate and sequentially compose LPO w.r.t. an "interface" specifying to which parts of a previous LPO a subsequent LPO is appended. Such an interface is given through an LPO I connecting events in the previous LPO to minimal events in the subsequent LPO. The composition w.r.t. to such an interface I is denoted by $*_I$ resp. $;_I$ and is realized w.r.t. the ordering given by I.



$$\alpha = A_1 + ((A_2 \,;\, (A_3)^{*_R});_S A_4)$$

Fig. 6. Composed run with partial iteration representing $L(N_1, m_1)$.

Figure 6 shows, how by this method the language $L(N_1, m_1)$ from Figure 2 can be specified: The LPO A_3 is iterated through appending it only to the a-labeled event and finally A_4 also is appended only to the a-labeled event. Note that it is in principle also possible to represent $L(N_2, m_2)$ through $A_1; (A_1 \parallel A_1)*_{A_1;(A_1\|A_1)}$. But the interpretation of this expression is not totally clear because there are two possibilities to use the interface $A_1; (A_1 \parallel A_1)$ to iterate $A_1 \parallel A_1$. One interpretation is that only one of the possibilities can be applied, another is that both possibilities can be applied in parallel (and only in this second case $L(N_2, m_2)$ is represented).



$$X = A_2 \,;\, ((X + A_1) \| A_1)$$
$$\alpha = A_1 + X$$

Fig. 7. Composed term with recursion representing $L(N_1, m_1)$.

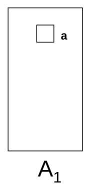
2.3 Recursion

Another possibility to generalize composed runs is to equip them with recursion. Through recursion it is possible specify that some behavior is repeated at certain points of a composed run. For this also variables can be used in a composed run. Each variable represents a set of LPOs. The set of LPOs specified through a variable X is given through an equation $X = \alpha(X)$, where $\alpha(X)$ is a composed run including X (X need not be minimal in $\alpha(X)$). The interpretation of such an equation is, that each occurrence of X on the right side may be replaced by the empty LPO or by $\alpha(X)$ and so on. It is in general also possible that there are more variables in one composed run and that there are more equations.

Figure 7 shows, how by this method the language $L(N_1, m_1)$ from Figure 2 can be given. Figure 8 shows, how by this method the language $L(N_2, m_2)$ from Figure 3 can be given.

3 Synthesis

The general ideas of region based synthesis of p/t-nets from partial languages $\mathcal{L}$ are as follows: The set of transitions of the synthesized net is the finite set of labels of $\mathcal{L}$. Places are defined by their initial marking and the weights on the arcs connecting them to transitions. Two kinds of places can be distinguished. In the case that there is an LPO specified in $\mathcal{L}$ which is no run of the net which has only the one considered place, this place restricts the behaviour too much. Such places are *non-feasible (w.r.t. $\mathcal{L}$)*. In the other case, the considered place is *feasible (w.r.t. $\mathcal{L}$)*. The aim is to add enough feasible places in order exactly reproduce the specified behavior.



$X = A_1 ; (X \,||\, X)$
$\alpha = X$

Fig. 8. Composed term with recursion representing $L(N_2, m_2)$.

Feasible places are computed through so called token flow regions which are defined on the level of the partial language [LJ06]: If two events x and y satisfy $x < y$ in an LPO $\mathrm{lpo} = (V, <, l) \in \mathcal{L}$, this specifies that the corresponding transitions $l(x)$ and $l(y)$ may be causally dependent. Such a causal dependency arises exactly if the occurrence of the transition $l(x)$ produces one or more tokens in a place, and some of these tokens are consumed by the occurrence of the other transition $l(y)$. Such a place can be defined as follows: Assign to every edge (x, y) of an LPO in $\mathcal{L}$ a natural number $r(x, y)$ representing *the number of tokens which are produced by the occurrence of $l(x)$ and consumed by the occurrence of $l(y)$ in the place to be defined*. For this, we extend each LPO $\mathrm{lpo} \in L$ by an initial event v_{lpo} and a final event, representing transitions producing the initial marking and consuming the final marking (after the occurrence of lpo). A feasible place p_r is then defined by assigning for each extended LPO $\mathrm{lpo} = (V, <, l) \in L$ a natural number $r(x, y)$ to each edge (x, y) function r, where it holds that (IN): $In(y, r) = \sum_{x <^\star y} r(x, y) = \sum_{x <^\star z} r(x, z) = In(z, r)$ for $l(y) = l(z)$, (OUT): $Out(y, r) = \sum_{y <^\star x} r(y, x) = \sum_{z <^\star x} r(z, x) = Out(z, r)$ for $l(y) = l(z)$ and $(INIT)$: $Out(v_{\mathrm{lpo}_1}, r) = Out(v_{\mathrm{lpo}_2}, r)$ for $\mathrm{lpo}_1, \mathrm{lpo}_2 \in \mathcal{L}$. We call $In(y, r)$ the *intoken flow* of y which is interpreted as the weight of the arc connecting the new place p_r with the transition $l(y)$ (i.e. $W(p_r, l(y)) = In(y, r)$). We call $Out(y, r)$ the *outtoken flow* of x, which is interpreted as the weight of the arc connecting the transition $l(x)$ with the new place p_r (i.e. $W(l(y), p_r) = Out(y, r)$). The outtoken flow of v_{lpo} is called *initial flow* and is interpreted as the initial marking of the new place p_r (i.e. $m_0(p_r) = Out(z, r)$). The value $r(x, y)$ is called the *token flow* between x and y.

A function r satisfying (IN), (OUT) and $(INIT)$ is called *region*. The main result of [LJ06] is that the set of places corresponding to regions of a partial language equals the set of feasible places w.r.t. this partial language.

This notion of regions can easily be adapted to each of the proposed finite representations. Namely, in each case a token flow function r need to fulfil requirements additional to (IN), (OUT) and $(INIT)$. In case, a partial language is given by a finite set of LPOs and an equivalence relation on prefixes of those LPOs, we require that

- r satisfies (IN), (OUT) and $(INIT)$ on the finite set of LPOs.
- r satisfies that for equivalent prefixes the sum of token flows on edges leaving one prefix equals the sum of token flows on edges leaving the other prefix.

In case, a partial language is given by a composed run using partial iteration, we require the same properties as for composed runs introduced in [LBDM08]. There an additional requirement was introduced for the so called set of iterated LPOs postulating that the initial and the final token flow of such iterated LPOs should be equal. The only difference now is that the initial and final token flow of such LPOs is computed in another way, namely w.r.t. the given interface. In case, a partial language is given by a composed run equipped with recursion equations, we require

- the same as for composed runs and additionally that
- for each equation $X = \alpha(X)$ the intial flow of $\alpha(X)$ equals the sum of token flows on edges ingoing an occurrence of X in $\alpha(X)$ for each such occurrence.

All these additional requirements can be represented as homogenous linear inequations as it is the case for (IN), (OUT) and $(INIT)$. Thus effective solution algorithms can be adapted.

References

[LBDM07] LORENZ, R. ; BERGENTHUM, R. ; DESEL, J. ; MAUSER, S.: Synthesis of Petri Nets from Finite Partial Languages. In: *ACSD*, IEEE Computer Society, 2007, S. 157–166

[LBDM08] LORENZ, R. ; BERGENTHUM, R. ; DESEL, J. ; MAUSER, S.: Synthesis of Petri Nets from Infinite Partial Languages. In: *Proceedings of ACSD*, 2008, S. 170 – 179

[LJ06] LORENZ, R. ; JUHÁS, G.: Towards Synthesis of Petri Nets from Scenarios. In: DONATELLI, S. (Hrsg.) ; THIAGARAJAN, P. S. (Hrsg.): *ICATPN* Bd. 4024, Springer, 2006 (Lecture Notes in Computer Science), S. 302–321

[LMB07] LORENZ, R. ; MAUSER, S. ; BERGENTHUM, R.: Theory of Regions for the Synthesis of Inhibitor Nets from Scenarios. In: KLEIJN, J. (Hrsg.) ; YAKOVLEV, A. (Hrsg.): *ICATPN* Bd. 4546, Springer, 2007 (Lecture Notes in Computer Science), S. 342–361