Faster Unfolding of General Petri Nets Based on Token Flows

Robin Bergenthum, Robert Lorenz, and Sebastian Mauser *

Department of Applied Computer Science,
Catholic University of Eichstätt-Ingolstadt,
firstname.lastname@ku-eichstaett.de

Abstract. In this paper we propose two new unfolding semantics for general Petri nets combining the concept of prime event structures with the idea of token flows developed in [11]. In contrast to the standard unfolding based on branching processes, one of the presented unfolding models avoids to represent isomorphic processes while the other additionally reduces the number of (possibly non-isomorphic) processes with isomorphic underlying runs. We show that both proposed unfolding models still represent the complete partial order behavior. We develop a construction algorithm for both unfolding models and present experimental results. These results show that the new unfolding models are much smaller and can be constructed significantly faster than the standard unfolding.

1 Introduction

Non-sequential Petri net semantics can be classified into unfolding semantics, process semantics, step semantics and algebraic semantics [17]. While the last three semantics do not provide semantics of a net as a whole, but specify only single, deterministic computations, unfolding models are a popular approach to describe the complete behavior of nets accounting for the fine interplay between concurrency and nondeterminism.

To study the behavior of Petri nets primarily two models for unfolding semantics were retained: labeled occurrence nets and event structures. In this paper we consider general Petri nets, also called place/transition Petri nets or p/t-nets (Figure 1). The standard unfolding semantics for p/t-nets is based on the developments in [19, 5] (see [14] for an overview) in terms of so called branching processes, which are acyclic occurrence nets having events representing transition occurrences and conditions representing tokens in places. Branching processes allow events to be in conflict through branching conditions. Therefore branching processes can represent alternative processes simultaneously (processes are finite branching processes

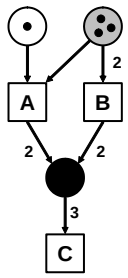

Fig. 1. Example net N. Instead of place-names we used different colors.

* Supported by the project SYNOPS of the German research council

without conflict). Branching processes were originally introduced in [19] for safe nets, and extended in [5] to initially one marked p/t-nets without arc weights, by viewing the tokens as individualized entities. In contrast to [5], branching processes for p/t-nets even individualize tokens having the same "history", i.e. several (concurrent) tokens produced by some transition occurrence in the same place are distinguished through different conditions (see [14]). Analogously as in [5] one can define a single maximal branching process, called the unfolding of the system (in the rest of the paper we will refer to this unfolding as the standard unfolding). The unfolding includes all possible branching processes as prefixes, and thus captures the complete non-sequential branching behavior of the p/t-net. In the case of bounded nets, according to a construction by McMillan [16] a complete finite prefix of the unfolding preserving full information on reachable markings can always be constructed. This construction was generalized in [3] to unbounded nets through building equivalence classes of reachable markings. In the case of bounded nets, the construction of unfoldings and complete finite prefixes is well analyzed and several algorithmic improvements are proposed in literature [7, 15, 13]. By restricting the relations of causality and conflict of a branching process to events, one obtains a labeled prime event structure [20] underlying the branching process, which represents the causality between events of the branching process. An event structure underlying a process, i.e. without conflict, is called a run. In the view of the development of fast model-checking algorithms employing unfoldings resp. event structures [6] there is still the important problem of efficiently building them.

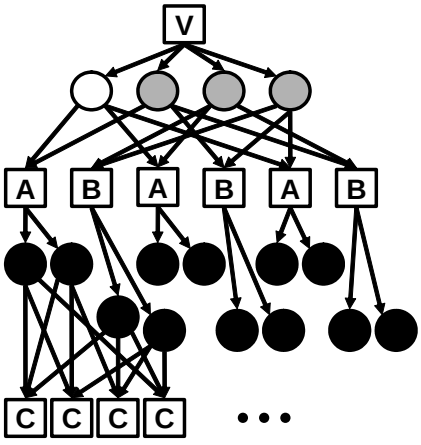

Fig. 2. Standard unfolding of N. The colors of conditions refer to the place the corresponding tokens belong to.

The p/t-net shown in Figure 1 has a finite standard unfolding (as defined for example in [14]). A part of this unfolding is shown in Figure 2. An unfolding has a unique minimal event producing the initial conditions. Each condition in the unfolding corresponds to a token in the net, i.e. tokens are individualized. In the initial marking there are three possibilities for transition B to consume two tokens from the grey place (and for transition A to consume one token from the grey and one token from the white place). All these possibilities define Goltz-Reisig processes [8] of the net, are in conflict and are reflected in the unfolding. That means, individualized tokens cause the unfolding to contain events with the same label, being in conflict and having the same number of equally labeled pre-conditions with the same "history" (for each place label), where two conditions have the same "history" if they have the same pre-event. Such events are called *strong identical*. In Figure 2 all A-labeled and all B-labeled events are strong identical, since all grey conditions have the same "history". Strong identical events produce isomorphic processes in the unfolding and therefore are redundant.

After the occurrence of transitions A and B there are four tokens in the black place and there are four possibilities for transition C to consume three of these tokens (Figure 2). For each of those possibilities, a C-labeled event is appended to the branching process. Two of these events consume two tokens produced by A and one token produced by B (these are strong identical), the other two consume one token produced by A and two tokens produced by B (these are also strong identical). The first pair of strong identical C-events is not strong identical to the second pair, but they all causally depend on the same two events. Such events having the same label, being in conflict and depending causally from the same events, are called *weak identical*. Weak identical events produce processes with isomorphic underlying runs and therefore also are redundant. Note finally, that the described four weak identical C-labeled events are appended to each of the three consistent pairs of A- and B-labeled events. That means, the individualized tokens in the worst case increase the number of events exponentially for every step of depth of the unfolding.

Figure 3 illustrates the labeled prime event structure underlying the unfolding shown in Figure 2. Formally a prime event structure is a partially ordered set of events (transition occurrences) together with a set of (so called) *consistency sets* [20]. "History-closed" (left-closed) consistency sets represent partially ordered runs. The labeled prime event structure underlying an unfolding is obtained by omitting the conditions and keeping the causality and conflict relations between events. Events not being in conflict define consistency sets. Thus, left-closed consistency sets correspond to processes and their underlying runs in the unfolding. Strong and weak identical events lead to consistency sets corresponding to processes with isomorphic underlying runs in the prime event structure.

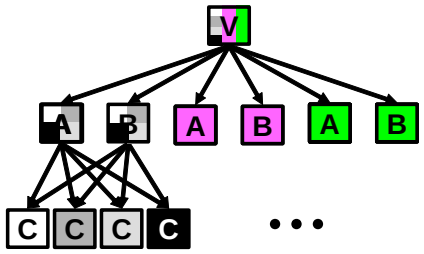

Fig. 3. Prime event structure of N. The colors of the events refer to the consistency sets. Transitive arcs are omitted.

In this paper we are interested in algorithms to efficiently construct unfoldings. As explained, the standard unfolding has the drawback, that it contains a lot of redundancy in form of isomorphic processes and processes with isomorphic underlying runs. This is caused by the individualization of tokens producing strong and weak identical events. Unfolding models with less nodes could significantly decrease the construction time, because a construction algorithm in some way always has to test all co-sets of events or conditions of the so-far constructed model to append further events. In this paper we propose two unfolding approaches reducing the number of events in contrast to the standard unfolding by neglecting (strong resp. weak) identical events.

Instead of considering branching processes, we use labeled prime event structures and assign so called token flows to its edges. Token flows were developed in [11] for a compact representation of processes. Token flows abstract from the individuality of conditions of a process and encode the flow relation of the process by natural numbers. For each place natural numbers are assigned to the edges of the partially ordered run

underlying a process. Such a natural number assigned to an edge (e, e') represents the number of tokens produced by the transition occurrence e and consumed by the transition occurrence e' in the respective place. This principle is generalized to branching processes/unfoldings and their underlying prime event structures in this paper.

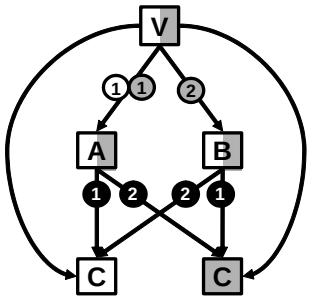

Fig. 4. Token flow unfolding of N. The coloring of the events illustrates the sets of consistent events.

The idea is to annotate each edge of the prime event structure underlying a branching process by the numbers of conditions between the corresponding pair of events of the branching process and omit isomorphic consistency sets having equal annotated token flow. The resulting prime event structure is shown in Figure 4. The event v is the unique initial event producing the initial marking. The edges have attached natural numbers, which are interpreted as token flows as described, where the colors refer to the places the tokens belong to. The assigned token flow specifies in particular that transition A consumes one initial token from the white place and one initial token from the grey place, while transition B consumes two initial tokens from the grey place. That means in this model the different possibilities for transition A and B of consuming initial tokens are not distinguished. Transition C either consumes one token produced by A and two tokens produced by B or vice versa in the black place. The respective two C-labeled events having the same pre-events but a different token flow are distinguished. They are in conflict yielding different consistency sets. In this approach strong identical events are avoided, while weak identical events still exist. Figure 4 only contains one of the three A and B events and two of the twelve C events. However, full information on reachable markings is still available. For example, the sum of all token flows assigned to edges from the initial event v to consistent events equals the initial marking. The example shows that through abstracting from the individuality of conditions, it is possible to generate an unfolding in form of a prime event structure with assigned token flow information having significantly less events than the standard unfolding.

A prime event structure with assigned token flow information is called a *token flow unfolding* if left-closed consistency sets represent processes and there are no strong identical events which are in conflict. Observe that to represent all possible processes we have to allow strong identical events which are not in conflict. For a given marked p/t-net, it is possible to define a unique maximal token flow unfolding, in which each process is represented through a consistency set with assigned token flows corresponding to the process. Figure 4 shows the maximal token flow unfolding for the example net N. We will show that the maximal token flow unfolding contains isomorphic processes only in specific situations involving auto-concurrency.

The token flow unfolding from Figure 4 still contains processes (consistency sets) which have isomorphic underlying runs, since token flow unfoldings still allow for weak

identical events. In Figure 5 a prime event structure with assigned token flow information is shown without weak identical events. Namely, the two weak identical C-labeled events in Figure 4 do not occur in Figure 5. This causes that the token flow information is not any more complete in contrast to token flow unfoldings, i.e. not each possible token flow distribution resp. process is represented. Instead example token flows are stored for each partially ordered run, i.e. each run is represented through one possible process. Note that still in this reduced unfolding full information on reachable markings is present, since markings reached through occurrence of a run do not depend on the token flow distribution within this run.

If a prime event structure with assigned token flow information does not contain weak identical events, this unfolding model is called a *reduced token flow unfolding*. We can define a unique maximal reduced token flow unfolding, in which each run is represented through a left-closed consistency set with assigned token flows corresponding to a process having this underlying run. It can be seen as a very compact unfolding model capturing the complete behavior of a p/t-net.

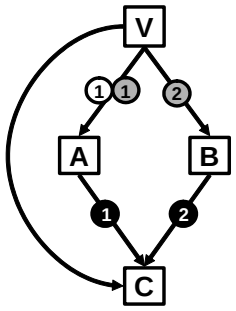

Fig. 5. Reduced token flow unfolding.

Figure 5 shows the maximal reduced token flow unfolding for the example net N. We will show that the maximal reduced token flow unfolding contains processes with isomorphic underlying runs only in specific situations involving auto-concurrency.

For both new unfolding approaches we develop a construction algorithm for finite unfoldings and present an implementation together with experimental results. Token flow unfoldings can be constructed in a similar way as branching processes. The main difference is that processes are not implicitly given through events being in conflict, but are explicitly stored in consistency sets. This implies that new events are appended to consistency sets and not to co-sets of conditions. From the token flow information we can compute, how many tokens in which place, produced by some event, are still not consumed by subsequent events. These tokens can be used to append a new event. The crucial advantage of token flow unfoldings is that much less events must be appended. One disadvantage is that a possible exponential number of consistency sets must be stored. Moreover, for the construction of the reduced token flow unfolding not the full token flow information is available, since not each possible but only one example token flow distribution is displayed. Therefore the procedure of appending a new event is more complicated, because eventually an alternative token flow distribution has to be calculated (there is an efficient method for this calculation based on the ideas in [11]). Experimental results show that the two new unfolding models can be constructed much faster and memory consumption is decreased. The bigger the markings and arc weights are, the more efficient is the new construction compared to the standard one.

Altogether, the two new unfolding approaches on the one hand allow a much more efficient construction, and on the other hand still offer full information on concurrency, nondeterminism, causality and reachable markings. In particular, the assigned token

flows allow to compute the reachable marking corresponding to a consistency set. This allows to apply the theory of complete finite prefixes of the standard unfolding also to the presented new models. Acceleration of model checking algorithms working on the standard unfolding can be done by adapting them to the new smaller unfolding models. Another benefit is, that the new methods may lead to a more efficient computation of the set of all processes of a p/t-net.

There are also other attempts to extend the unfolding approach of [19] for safe nets to p/t-nets, where in some of them tokens are individualized as in the standard unfolding ([17, 18]) and in some of them such an individualization of tokens is avoided as in our approach ([10, 9, 2, 12, 1]). In [17, 18] conditions are grouped into families (yielding so called decorated occurrence nets) in order to establish desirable algebraic and order-theoretic properties. In [10] so called local event structures instead of prime event structures are introduced as an unfolding semantics of p/t-nets without autoconcurrency. In this approach, conflict and causality relations among events are not any more explicitly given by the model structure. Algorithmic aspects are not considered. In [2] arbitrarily valued and non-safe occurrence nets are used. Also here the direct link with prime event structures is lost. In [9], general nets are translated into safe nets by introducing places for reachable markings (which considerably increases the size of the structure) in order to apply branching processes for safe nets. In [1] a swapping equivalence between conditions introduces a collective token view.[1] Finally, in [12] events and conditions are merged according to appropriate rules yielding a more compact unfolding structure which not longer needs to be acyclic. Nevertheless it can be used for model checking. It cannot be directly computed from the p/t-net but only from its finite complete prefix which needs to be computed first. In contrast to all these approaches we propose a compact unfolding semantics avoiding individualized tokens while still explicitly reflecting causal relations between events through prime event structures. Moreover, basic order theoretic properties such as the existence of a unique maximal unfolding can be established. Our main focus is the development of fast construction algorithms, established so far for the finite case.

The remainder of the paper is organized as follows: In Section 2 we introduce basic mathematical notations and briefly restate the standard unfolding approach for p/t-nets. In Section 3 we develop the two new unfolding models. We prove that in both cases there is a unique maximal unfolding representing all processes resp. runs and formalize in which cases isomorphic processes resp. runs are avoided. Finally, in Section 4 we present algorithms for the construction of the new unfolding models in the finite case and provide experimental results in Section 5.

2 P/T-Nets and Standard Unfolding Semantics

In this section we recall the definitions of place/transition Petri nets and the standard unfolding semantics based on branching processes. We begin with some basic mathematical notations.

[1] Note here that the token flow unfolding and the reduced token flow unfolding define an equivalence on processes which is finer than the swapping equivalence, i.e. weak and strong equivalent events always produce swapping equivalent processes.

We use $\mathbb{N}$ to denote the *nonnegative integers*. A *multi-set* over a set A is a function $m : A \to \mathbb{N} \in \mathbb{N}^A$. For an element $a \in A$ the number $m(a)$ determines the number of occurrences of a in m. Given a binary relation $R \subseteq A \times A$ over a set A, the symbol R^+ denotes the *transitive closure* of R and R^* denotes the *reflexive transitive closure* of R. A *directed graph* G is a tuple $G = (V, \to)$, where V is its set of *nodes* and $\to \subseteq V \times V$ is a binary relation over V called its set of *arcs*. As usual, given a binary relation $\to$, we write $a \to b$ to denote $(a, b) \in \to$. For $v \in V$ and $W \subseteq V$ we denote by ${}^\bullet v = \{v' \in V \mid v' \to v\}$ the *preset* of v, and by $v^\bullet = \{v' \in V \mid v \to v'\}$ the *postset* of v, ${}^\bullet W = \bigcup_{w \in W} {}^\bullet w$ is the *preset* of W and $W^\bullet = \bigcup_{w \in W} w^\bullet$ is the *postset* of W.

A *partial order* is a directed graph $(V, <)$, where $< \subseteq V \times V$ is an irreflexive and transitive binary relation. In this case, we also call $<$ a partial order. In the context of this paper, a partial order is interpreted as "earlier than"-relation between events. Two nodes (events) $v, v' \in V$ are called *independent* if $v \not< v'$ and $v' \not< v$. 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 $S \subseteq V$ fulfilling $\forall x, y \in S : \ x \,\mathrm{co}_< \, y$. A *cut* is a maximal co-set. For a co-set S and a node $v \in V \setminus S$ we write $v < S$ ($v > S$), if $\exists s \in S : \ v < s$ ($\exists s \in S : \ v > s$), and $v \,\mathrm{co}_< S$, if $\forall s \in S : \ v \,\mathrm{co}_< s$. A node v is called *maximal* if $v^\bullet = \emptyset$, and *minimal* if ${}^\bullet v = \emptyset$. A subset $W \subseteq V$ is called *left-closed* if $\forall v, v' \in V : \ (v \in W \wedge v' < v) \Longrightarrow v' \in W$. For a left-closed subset $W \subseteq V$, the partial order $(W, <|_{W \times W})$ is called *prefix* of $(V, <)$, defined by W. The *left-closure* of a subset W is given by the set $W \cup \{v \in V \mid \exists w \in W : \ v < w\}$. The node set of a finite prefix equals the left-closure of the set of its maximal nodes. 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 $<_1 \subseteq <_2$. By $<_s \subseteq <$ we denote the smallest subset $<'$ of $<$ which fulfils $(<')^+ = <$, called the *skeleton of* $<$.

A *labeled partial order* (LPO) is a triple $(V, <, l)$, where $(V, <)$ is a partial order, and l is a *labeling function* on V. We use all notations defined for partial orders also for LPOs. If V is a set and $l : V \to X$ is a labeling function on V, then for a finite subset $W \subseteq V$, we define the multi-set $l(W) \subseteq \mathbb{N}^X$ by $l(W)(x) = |\{v \in W \mid l(v) = x\}|$. LPOs are used to represent partially ordered runs of Petri nets. Such runs are distinguished only up to isomorphism. Two LPOs $(V_1, <_1, l_2)$ and $(V_1, <_1, l_2)$ are *isomorphic* if there is a bijective mapping $\varphi : V_1 \to V_2$ satisfying $\forall v_1 \in V_1 : \ l(v_1) = l(\varphi(v_1))$ and $\forall v_1, v_1' \in V_1 : \ v_1 <_1 v_1' \Longleftrightarrow \varphi(v_1) <_2 \varphi(v_1')$.

A *net* is a triple $N = (P, T, F)$, where P is a set of *places*, T is a set of *transitions*, satisfying $P \cap T = \emptyset$, and $F \subseteq (P \cup T) \times (T \cup P)$ is a *flow relation*. Places and transitions are called the nodes of N. Presets and postsets of (sets of) nodes are defined w.r.t. the directed graph $(P \cup T, F)$. We denote $\preceq_N = F^*$ and $\prec_N = F^+$. If N is clear from the context, we also write $\preceq$ instead of $\preceq_N$ and $\prec$ instead of $\prec_N$.

Assume now that $\prec_N = \prec$ is a partial order. Then two nodes x, y (places or transitions) of N are *in conflict*, denoted by $x \# y$, if there are distinct transitions $t, t' \in E$ with ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ such that $t \preceq x$ and $t' \preceq y$. Two nodes x, y are called *independent* if $x \,\mathrm{co}_\prec y$ and $\neg(x \# y)$. Maximal and minimal nodes of N and prefixes of N are defined w.r.t. $(P \cup T, \prec)$.

Definition 1 (Place/transition net). *A* place/transition-net *(shortly* p/t-net*)* N *is a quadruple* (P, T, F, W)*, where* (P, T, F) *is a net with finite sets of places and transitions, and* $W : F \to \mathbb{N} \setminus \{0\}$ *is a* weight function. *A* marking *of a p/t-net* $N = (P, T, F, W)$

is a function $m : P \to \mathbb{N}$. *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.

We extend the weight function W to pairs of net elements $(x, y) \in (P \times T) \cup (T \times P)$ satisfying $(x, y) \notin F$ by $W((x, y)) = 0$. A transition $t \in \mathbb{N}$ is *enabled to occur in a marking* m of N if $\forall p \in P : \ m(p) \geq W((p, t))$. If t is enabled to occur in a marking m, then its *occurrence* leads to the new marking m' defined by $m'(p) = m(p) - W((p, t)) + W((t, p))$ for $p \in P$.

Unfolding semantics of p/t-nets is given by so called *branching processes* which are based on occurrence nets. A conflict relation between events distinguishes alternative runs. Runs are given by conflict-free, left-closed sub-nets of branching processes.

Definition 2 (Occurrence net). *An* occurrence net *is a net* $O = (B, E, G)$ *satisfying*

- O *is acyclic, i.e.* $\prec_O$ *is a partial order.*
- $\forall b \in B : \ |{}^\bullet b| \leqslant 1$.
- $\forall x \in B \cup E : \ \neg(x \# x)$.
- $\forall x \in B \cup E : |\{y \mid y \prec x\}|$ *is finite.*

Elements of B are called *conditions* and elements of E are called *events*. $MIN(O)$ denotes the set of minimal elements (w.r.t. $\prec_O$).

Definition 3 (Branching process). *Let* (N, m_0)*,* $N = (P, T, F, W)$ *be a marked p/t-net. A* branching process *of* (N, m_0) *is a pair* $\pi = (O, \rho)$ *where* $O = (B, E, G)$ *is an occurrence net and* $\rho : B \cup E \to X$ *with* $P \cup T \subset X$ *is a labeling function satisfying:*

- *There is* $e_{init} \in E$ *with* $MIN(O) = \{e_{init}\}$ *and* $\rho(e_{init}) \notin P \cup T$.
- $\forall b \in B : \ \rho(b) \in P$ *and* $\forall e \in E \setminus \{e_{init}\} : \ \rho(e) \in T$.
- $\forall e \in E \setminus \{e_{init}\}, \forall p \in P : \ |\{b \in {}^\bullet e \mid \rho(b) = p\}| = W((p, \rho(e))) \wedge |\{b \in e^\bullet \mid \rho(b) = p\}| = W((\rho(e), p))$.
- $\forall p \in P : \ |\{b \in e^\bullet_{init} \mid \rho(b) = p\}| = m_0(p)$.
- $\forall e, f \in E : \ ({}^\bullet e = {}^\bullet f \wedge \rho(e) = \rho(f)) \Longrightarrow (e = f)$.

In a branching process, $\prec$ is interpreted as "earlier than"-relation between transition occurrences. A finite branching process with empty conflict relation is called a *process*.

Two branching processes $\pi' = (O', \rho')$, $O' = (B', E', G')$, and $\pi = (O, \rho)$, $O = (B, E, G)$, are isomorphic, if there is a bijection $Iso : B \cup E \to B' \cup E'$ satisfying $Iso(B) = B'$, $Iso(E) = E'$, $\rho' \circ Iso = \rho$ and $(x, y) \in G \Leftrightarrow (Iso(x), Iso(y)) \in G'$ for $x, y \in B \cup E$.

A branching process $\pi = (O, \rho)$, $O = (B, E, G)$, is a prefix of another branching process $\pi' = (O', \rho')$, $O' = (B', E', G')$, denoted by $\pi \sqsubseteq \pi'$, if O is a prefix of O' satisfying $B = MIN(O) \cup (\bigcup_{e \in E} e^\bullet)$ and ρ is the restriction of ρ' to $B \cup E$. For each marked p/t-net (N, m_0) there exists a unique, w.r.t. $\sqsubseteq$ maximal, branching process $\pi_{max}(N, m_0)$, called the *unfolding of* (N, m_0).

Sometimes one is only interested in storing the causal dependencies of events of a branching process. For this conditions are omitted and the $\prec$- and $\#$-relation are kept for events. Formally the resulting object is a so-called *prime event structure*.

Definition 4 (Prime event structure). *A* prime events structure *is a triple* $PES = (E, Con, \prec)$ *consisting of a set* E *of events, a partial order* $\prec$ *on* E *and a set* Con *of finite subsets of* E *satisfying:*

- $\forall e \in E : \{e' \mid e' \prec e\}$ *is finite.*
- $\{e\} \in Con$*.*
- $Y \subseteq X \in Con \Longrightarrow Y \in Con$*.*
- $((X \in Con) \wedge (\exists e' \in X : e \prec e')) \Longrightarrow (X \cup \{e\} \in Con)$*.*

A consistent subset *of* E *is a subset* X *satisfying* $\forall Y \subseteq X, Y$ *finite* $: Y \in Con$*. The conflict relation* $\#$ *between events of* PES *is defined by* $e \# e' \Leftrightarrow \{e, e'\} \notin Con$*.*

A pair (PES, l)*, where* PES *is a prime events structure and* l *is a labeling function on* E*, is called* labeled prime event structure.

A (labeled) prime event structure with E consistent we interpret as an LPO, i.e. in this case we omit the set of consistency sets Con.

If $\pi = (O, \rho)$, $O = (B, E, G)$ is a branching process, then $PES(\pi) = (E, Con, \prec |_{E \times E})$, where $X \in Con$ if and only if $X \subseteq E$ is finite and fulfills $\forall e, e' \in X : \neg(e \# e')$, is a prime event structure. $PES(\pi)$ is called *corresponding to* π. If π is a process, then $PES(\pi)$ is a finite LPO, called the *run underlying* π.

3 Unfoldings Based on Token Flows

One basic problem of the unfolding of a p/t-net is, that it contains a lot of redundancy. This arises from the individuality of conditions in branching processes. When appending a new transition occurrence to a branching process, each particular choice of a set of conditions representing the preset of this transition yields a different process, where some of these processes are isomorphic and others have isomorphic underlying runs (see Figure 2 and the explanations in the introduction). In this section we propose two new unfolding semantics of p/t-nets avoiding such redundancy. Both approaches are based on the notion of token flows presented in [11]. In the following we restate this notion and its role in the representation of single processes. In the next subsection, the concepts will be transferred to unfoldings.

In the following, LPOs are considered to be finite. The edges of LPOs, representing partially ordered runs of a p/t-net, are annotated by (tuples of) non-negative integers interpreted as token flow between transition occurrences. Namely, for a process $K = (O, \rho)$, $O = (B, E, G)$, of a marked p/t-net (N, m_0), $N = (P, T, F, W)$, we defined a so called *canonical token flow function* $x_K : \prec \rightarrow \mathbb{N}^P$ assigned to the edges of the run $(E, \prec, \rho)$ underlying K via $x_K((e, e')) = \rho(e^\bullet \cap {}^\bullet e')$. That means $x_K((e, e'))$ represents for each place the number of tokens which are produced by the occurrence of the transition $\rho(e)$ and then consumed by the occurrence of $\rho(e')$. Such a token flow function abstracts away from the individuality of conditions in a process and encodes the token flow by natural numbers for each place. It is easy to see that x_K satisfies:

- (IN): $\forall e \in E \setminus \{e_{init}\}, \forall p \in P : (\sum_{e' \prec e} x_K(e', e))(p) = W(p, \rho(e))$.
- (OUT): $\forall e \in E \setminus \{e_{init}\}, \forall p \in P : (\sum_{e \prec e'} x_K(e, e'))(p) \leqslant W(\rho(e), p)$.
- (INIT): $\forall p \in P : (\sum_{e_{init} \prec e'} x(e_{init}, e'))(p) \leqslant m_0(p)$.

– (MIN): $\forall (e, e') \in \prec_s : \ (\exists p \in P : \ x_K(e, e')(p) \geqslant 1)$.

(IN), (OUT) and (INIT) reflect the consistency of the token flow distribution given by x_K with the initial marking and the arc weights of the considered net. (MIN) holds since skeleton arcs define the "earlier than"-causality between transition occurrences and this causality is produced by non-zero token flow. Non-skeleton edges may carry a zero token flow, since they are induced by transitivity of the partial order. A zero flow of tokens means, that there is no direct dependency between events. In particular, there are no token flows between concurrent events.

In [11] we showed, that the other way round for an LPO $(E, \prec, \rho)$ with unique initial event e_{init}, a token flow function $x : \prec \rightarrow \mathbb{N}^P$ satisfying (IN), (OUT), (INIT) and (MIN) is a canonical token flow function of a process. That means processes are in one-to-one correspondence with LPOs having token flow assigned to their edges fulfilling (IN), (OUT), (INIT) and (MIN). Such LPOs yield an equivalent but more compact representation of partially ordered runs. In particular full information on reachable markings as well as causal dependency and concurrency among events is preserved.

3.1 Token Flow Unfolding

In the following, we extend these ideas to branching processes by assigning token flows to the edges of prime event structures. We will prove that in such a way one gets a more compact representation of the branching behavior of p/t-nets while preserving full information on markings, concurrency, causal dependency and conflict.

Let $PES = (E, Con, \prec)$ be a prime event structure. We denote the set of left-closed consistency sets by $Con_{pre} \subseteq Con$. For a *token flow function* $x : \prec \rightarrow \mathbb{N}^P$, a consistency set $C \in Con_{pre}$ and an event $e \in C$ we denote

– $IN_C(e) = \sum_{e' \prec e, \, e' \in C} x(e', e)$ the *intoken flow of* e *w.r.t.* C.
– $OUT_C(e) = \sum_{e \prec e', \, e' \in C} x(e, e')$ the *outtoken flow of* e *w.r.t.* C.

A prime token flow event structure is a labeled prime event structure together with a token flow function. Since equally labeled events represent different occurrences of the same transition, they are required to have equal intoken flow. Since not all tokens which are produced by an event are consumed by further events, there is no analogous requirement for the outtoken flow. It is assumed that there is a unique initial event producing the initial marking.

Definition 5 (Prime token flow event structure). *A* prime token flow event structure *is a pair* $((PES, l), x)$*, where* $PES = (E, Con, \prec)$ *is a prime event structure,* l *is a labeling function on* E *and* $x : \prec \rightarrow \mathbb{N}^P$ *is a* token flow function *satisfying:*

– *There is a unique minimal event* e_{init} *w.r.t.* $\prec$ *with* $l(e_{init}) \neq l(e)$ *for all* $e \neq e_{init}$.
– $\forall C, C' \in Con_{pre}, \forall e \in C, e' \in C' : \ l(e) = l(e') \Longrightarrow IN_C(e) = IN_{C'}(e')$.

A token flow unfolding of a marked p/t-net is a prime token flow event structure, in which intoken and outtoken flows are consistent with the arc weights resp. the initial

marking of the net within each left-closed consistency set. Moreover, we neglect so-called *strong identical events* in such unfoldings which turn out to produce isomorphic process nets. [2]

Definition 6 (Strong identical events). *Let $((PES, l), x)$ be a prime token flow event structure. Two events $e, e' \in E$ fulfilling*

$$(l(e) = l(e')) \wedge ({}^\bullet e = {}^\bullet e') \wedge (\forall f \in {}^\bullet e : \; x(f, e) = x(f, e'))$$

are called strong identical.

Definition 7 (Token flow unfolding). *Let (N, m_0), $N = (P, T, F, W)$, be a marked p/t-net. A* token flow unfolding of (N, m_0) *is a prime token flow event structure $((PES, l), x)$, $l : E \to X$ with $T \subset X$ and $\forall e \in E \setminus \{e_{init}\} : \; l(e) \in T$, satisfying:*

- *(IN):* $\forall C \in Con_{pre}, \forall e \in C \setminus \{e_{init}\}, \forall p \in P : \; IN_C(e)(p) = W(p, l(e))$.
  (OUT): $\forall C \in Con_{pre}, \forall e \in C \setminus \{e_{init}\}, \forall p \in P : \; OUT_C(e)(p) \leqslant W(l(e), p)$.
  (INIT): $\forall C \in Con_{pre}, \forall p \in P : \; OUT_C(e_{init})(p) \leqslant m_0(p)$.
  (MIN): $\forall (e, e') \in \prec_s : \; (\exists p \in P : \; x(e, e')(p) \geqslant 1)$.
- *There are no strong identical events e, e' satisfying $\{e, e'\} \notin Con$.*

Two token flow unfoldings $\mu' = ((PES', l'), x')$, $PES' = (E', Con', \prec')$, and $\mu = ((PES, l), x)$, $PES = (E, Con, \prec)$, are isomorphic *if there is a bijection $I : E \to E'$ satisfying $\forall e \in E : \; l(e) = l'(I(e)) \wedge I({}^\bullet e) = {}^\bullet I(e) \wedge I(e^\bullet) = I(e)^\bullet$, $\forall C \subseteq E : \; C \in Con \Leftrightarrow I(C) \in Con'$ and $\forall e \prec f : \; x(e, f) = x'(I(e), I(f))$.*

Given a token flow unfolding $\mu = ((PES, l), x)$, $PES = (E, Con, \prec)$, each non-empty left-closed subset $E' \subseteq E$ defines a token flow unfolding $\mu' = ((PES', l'), x')$, $PES' = (E', Con', \prec')$ by $Con' = \{C \in Con \mid C \subseteq E'\}$, $\prec' = \prec |_{E' \times E'}$, $l' = l|_{E'}$ and $x' = x|_{\prec'}$. Each token flow unfolding $\mu'' = ((PES'', l'), x')$, $PES'' = (E', Con'', \prec')$, $Con'' \subseteq Con'$ is called prefix of μ, denoted by $\mu'' \sqsubseteq \mu$. By $\sqsubset = \sqsubseteq \setminus id$, a partial order on the set of token flow unfoldings is given. We now want to prove that up to isomorphism, there exists a unique maximal (in general infinite) token flow unfolding w.r.t. $\sqsubset$. The partial order $\sqsubset$ can be defined through appending new events to (consistency sets of) existing token flow unfoldings starting with the initial event. There is a unique maximal token flow unfolding (fix point) only if the order of appending events does not matter, i.e. if events are appended in different orders, isomorphic token flow unfoldings are constructed.

A new transition occurrence can be appended to a consistency set, if there are enough *remaining* tokens produced by events in the consistency set (tokens which are not consumed by subsequent events in the consistency set). For $e \in E \setminus \{e_{init}\}$ and $C \in Con_{pre}$, the remaining tokens produces by e are formally given by the *residual token flow* $Res_C(e)$ (of e w.r.t. C) defined by $Res_C(e)(p) = W(l(e), p) - \sum_{e \prec e', \, e' \in C} x(e, e')(p)$ for $p \in P$. Similarly, for each $p \in P$, $Res_C(e_{init})(p) =$

[2] Note that omitting strong identical events disables the possibility of applying prime event structures having a set of consistency sets defined by a binary conflict relation, which is the case for example for prime event structures corresponding to branching processes.

$m_0(p) - \sum_{e_{init} \prec e',\, e' \in C} x(e_{init}, e')(p)$. Let $Mar(C) = \sum_{e \in C} Res_C(e)$ be the the *residual marking* of C. If there are enough tokens in the residual marking to fire a transition t, there may be several choices which of the remaining tokens are used to append a respective transition occurrence to the consistency set. Each such choice is formally represented by an *enabling function* $y : C \to \mathbb{N}^P$ satisfying $\forall e \in C : Res_C(e) \geqslant y(e)$ and $\forall p \in P : (\sum_{e \in C} y(e))(p) = W((p,t))$. Such an enabling function defines a new event e_y by $l'(e_y) = t$, ${}^\bullet e_y = \{e \in C \mid \exists e' : y(e') \neq 0 \wedge (e = e' \vee e \prec e')\}$ and $\forall e \in {}^\bullet e_y : x'(e, e_y) = y(e)$. If there is already a strong identical event not belonging to the consistency set, then this strong identical event is added to the consistency set. Otherwise, the new event is added.

Definition 8 (Appending events). *Let (N, m_0), $N = (P, T, F, W)$, be a marked p/t-net and let $\mu = ((PES, l), x)$, $PES = (E, Con, \prec)$, be a token flow unfolding of (N, m_0).*

Let $t \in T$ and $C \in Con_{pre}$ be such that $Mar(C)(p) \geqslant W((p,t))$ for each p. Let y be an enabling function and e_y be the associated new event.

If there is no event $e \notin C$ which is strong identical to e_y, then we define a prime token flow event structure $Ext(\mu, C, y, t) = ((PES', l'), x')$, $PES' = (E', Con', \prec')$, through $E' = E \cup \{e_y\}$, $l'|_E = l$, $x'|_\prec = x$, $\prec' |_{E \times E} = \prec$ and $Con' = Con \cup \{C' \cup \{e_y\} \mid C' \subseteq C\}$. We say that μ is extended by *e_y.*

If there is an event $e_{id} \notin C$ which is strong identical to e_y, then we define a prime token flow event structure $Ext(\mu, C, y, t) = ((PES', l'), x')$, $PES' = (E', Con', \prec')$, through $E' = E$, $l' = l$, $x' = x$, $\prec' = \prec$ and $Con' = Con \cup \{C' \cup \{e_{id}\} \mid C' \subseteq C\}$. We say that μ is updated by *e_y.*

The following lemma ensures that we have defined an appropriate procedure to append events:

Lemma 1. *$Ext(\mu, C, y, t)$ fulfills:*

(i) *$Ext(\mu, C, y, t)$ is a token flow unfolding.*
(ii) *$\mu \sqsubseteq Ext(\mu, C, y, t)$.*
(iii) *Every finite token flow unfolding μ can be constructed by the procedure shown in Definition 8: Given a token flow unfolding $\mu = ((PES, l), x)$, $PES = (E, Con, \prec)$, there exists a sequence $\mu_0, \ldots, \mu_n$ of token flow unfoldings with $\mu_0 = ((\{e_{init}\}, \{\{e_{init}\}\}, \emptyset), id), \emptyset)$, $\mu_n = \mu$ and $\mu_{i+1} = Ext(\mu_i, C_i, y_i, t_i)$ for $i = 0, \ldots, n-1$.*

Proof. The first and second statement follow by construction. The third one can be shown as follows: Fix one ordering of $E = \{e_1, \ldots, e_n\}$, such that $e_i \prec e_j \Longrightarrow i < j$ and denote $E_i = E \setminus \{e_i, \ldots, e_n\}$. Then $\mu_0 \sqsubset \mu_{E_2} \sqsubset \ldots \sqsubset \mu_{E_n} \sqsubset \mu_E$, where $\mu_{E'}$ is the prefix of μ defined by a left-closed set $E' \subseteq E$. By definition, there are triples $(C_1^i, y_1^i, l(e_i)), \ldots, (C_m^i, y_m^i, l(e_i))$, such that $\mu_{E_{i+1}}$ can be constructed from μ_{E_i} by appending $l(e_i)$-occurrences in arbitrary order to C_j^i via y_j^i for $j = 1, \ldots, m$. Namely, $C_1^i, \ldots C_m^i$ are the sets arising by omitting e_i from every left-closed consistency set in $\mu_{E_{i+1}}$ which includes e_i and all y_j^i are defined according to the intoken flow of e_i. That means $supp_i = \{y_k^i > 0\} = \{y_j^i > 0\}$ and $y_k^i|_{supp_i} = y_j^i|_{supp_i}$ for all k, j. Therefore, actually in the first appending step the event e_i is appended and in the further $m - 1$ steps only consistency sets are updated.

In the construction of the above proof, the resulting token flow unfolding does not depend on the used ordering of the events in E and also does not depend on the used ordering of the consistency sets enabling a fixed event e. This means that extending finite token flow unfoldings by new events in different orders and w.r.t. different consistency sets leads to isomorphic token flow unfoldings if after each extension all consistency sets which enable the considered event are updated. Observe moreover that by definition also the extension by a new event and the update by another event can be mixed up. This gives the following statement:

Lemma 2. *Let (N, m_0) be a marked p/t-net. There is a token flow unfolding $Unf_{max}(N, m_0)$, which is maximal w.r.t. $\sqsubset$ (no more events can be appended to finite prefixes) and unique up to isomorphism.*

$Unf_{max}(N, m_0)$ can be defined as the limit of a sequence of finite token flow unfoldings $(\mu_n)_{n\in\mathbb{N}}$ with $\mu_{n+1} = Ext(\mu_n, C, y, t)$, since the order of appending events does not matter. Each finite left-closed consistency set C of $Unf_{max}(N, m_0)$ represents a process π_C of (N, m_0) in the sense that the LPO $\mathrm{lpo}_C = (C, \prec |_{C\times C}, l|_C)$ is the run underlying π_C and $x_C = x|_{C\times C}$ is the canonical token flow function of π_C (this follows from [11] since x_C satisfies (INIT), (IN), (OUT) and (MIN) on lpo_C). Moreover, in $Unf_{max}(N, m_0)$ all processes of (N, m_0) are represented by finite left-closed consistency sets. Namely, for each process, the underlying run with assigned canonical token flow defines a token flow unfolding and without loss of generality we can assume that this token flow unfolding equals μ_k of a defining sequence of $Unf_{max}(N, m_0)$ for some k.

Theorem 1. *Let (N, m_0) be a marked p/t-net. Then for each process π of (N, m_0) there is a left-closed consistency set C of $Unf_{max}(N, m_0)$ such that π_C is isomorphic to π.*

To show that $Unf_{max}(N, m_0)$ avoids the generation of isomorphic processes, we prove that only in special auto-concurrency situations processes of (N, m_0) are represented more than once in $Unf_{max}(N, m_0)$.

Theorem 2. *Let (N, m_0) be a marked p/t-net and π be a finite process of (N, m_0). If in $Unf_{max}(N, m_0) = ((PES, l), x)$, $PES = (E, Con, \prec)$, there are two finite sets $C \neq C' \in Con_{pre}$ representing π, then there exist events $e \in C, e' \in C', e \neq e'$ such that e and e' are strong identical and fulfill $\{e, e'\} \in Con$.*

Proof. Assume there are finite $C, C' \in Con_{pre}$ such that the processes π_C and $\pi_{C'}$ are isomorphic. Let $e \in C \setminus C'$ with ${}^\bullet e \subseteq C \cap C'$. Such an event e exists since $C \cap C'$ defines a prefix of PES containing e_{init} and therefore e can be chosen as a minimal element w.r.t. $\prec$ in $C \setminus C'$. Let $e' \in C'$ be the image of e under the isomorphism relating π_C and $\pi_{C'}$. Since π_C and $\pi_{C'}$ are isomorphic, the left-closed consistency sets of all pre-events of e resp. e' define isomorphic processes. Thus, either e and e' are strong identical, or there are $f \in {}^\bullet e \setminus {}^\bullet e'$ and $f' \in {}^\bullet e' \setminus {}^\bullet e$ fulfilling the same property as e and e' that the left-closed consistency sets of all pre-events of f resp. f' define isomorphic processes. Since the number of pre-events of f and f' is smaller than the number of pre-events of e and e' (i.e. the procedure can only finitely often be iterated),

and e_{init} is a common pre-event, there is some pair of events g and g' being strong identical. By definition we have $g \in C$, $g' \in C'$ and $g \neq g'$. The definition of token flow unfoldings ensures $\{g, g'\} \in Con$ (since g, g' are strong identical).

Since e and e' are strong identical, they in particular have the same label and $\{e, e'\} \in Con$ shows that they can occur concurrently in some marking.

Corollary 1. *If (N, m_0) allows no auto-concurrency (i.e. there is no reachable marking m, such that there is a transition t fulfilling $\forall p \in P : m(p) \geq 2 \cdot W((p,t))$), there is a one-to-one correspondence between left-closed consistency sets of $Unf_{max}(N, m_0)$ and (isomorphism classes of) processes.*

Although we have seen that the non-existence of strong identical events is not enough to avoid isomorphic processes in general, the number of isomorphic processes represented in token flow unfoldings is significantly smaller than in the standard unfolding approach (see the experimental results). Figure 6 shows a situation as discussed in Theorem 2, where the token flow unfolding as introduced so far is not small enough to completely neglect isomorphic processes. Namely, the two B-labeled events produce two isomorphic processes, despite they are not strong identical (because they have no common pre-events). Observe however, that the two A-labeled pre-events of the two B-labeled events are themselves strong identical, but are not in conflict (they are concurrent). To avoid such situation, we must generalize the notion of strong identical events in the sense that two strong identical events have not necessarily common, but strong identical pre-events. This setting is formally described by the notion of isomorphic strong identical events as follows:

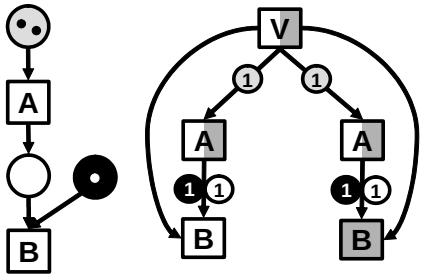

Fig. 6. P/t-net with token flow unfolding containing two isomorphic (maximal) processes.

Let $((PES, l), x)$ be a prime token flow event structure. Let $\cong \subseteq E \times E$ be the least equivalence relation satisfying for all $e, e' \in E$:

- $((l(e) = l(e')) \wedge ({}^\bullet e = {}^\bullet e') \wedge (\forall f \in {}^\bullet : x(f, e) = x(f, e'))) \Longrightarrow (e \cong e')$.
- $((l(e) = l(e')) \wedge (\exists I : {}^\bullet e \to {}^\bullet e'$ bijective $: (\forall f \in {}^\bullet e : f \cong I(f) \wedge x(f, e) = x(I(f), e'))) \Longrightarrow (e \cong e')$.

Two $\cong$-equivalent events e, e' are called *isomorphic strong identical*. Basically, omitting isomorphic strong identical events yields a token flow unfolding representing no isomorphic (maximal) processes at all (can be deduced similarly as Theorem 2). But considering such an approach we encountered several intricate technical problems. In particular, a test for the isomorphic strong identical property is complicated, such that the algorithmic applicability is questionable.

3.2 Reduced Token Flow Unfolding

In a token flow unfolding there is still redundancy w.r.t causality and concurrency, since there are consistency sets which induce processes which have the same underlying run (but a different token flow distribution). Such consistency sets are caused by so-called *weak identical events* (compare the introduction). To avoid weak identical events, since many different processes produce one run and token flow distributions correspond to processes, we store for each consistency set an example token flow distribution.

That means, we need to extend our model of prime event structures $PES = (E, Con, \prec)$ extended by token flows such that we can store for each consistency set $C \in Con_{pre}$ an individual token flow $x_C : \prec |_{C \times C} \to \mathbb{N}^P$. For such x_C and an event $e \in C$ we denote $IN_C(e) = \sum_{e' \prec e} x_C(e', e)$ and $OUT_C(e) = \sum_{e \prec e'} x_C(e, e')$. We introduce *generalized prime token flow event structures* as pairs $((PES, l), (x_C)_{C \in Con_{pre}})$, where $PES = (E, Con, \prec)$ is a prime event structure, l is a labeling function on E and $(x_C)_{C \in Con_{pre}}$ is a family of *token flow functions* $x_C : \prec |_{C \times C} \to \mathbb{N}^P$ satisfying analogous conditions as prime token flow event structures:

- There is a unique minimal event e_{init} w.r.t. $\prec$ with $l(e_{init}) \neq l(e)$ for all $e \neq e_{init}$.
- $\forall C, C' \in Con_{pre}, \forall e \in C, e' \in C' : \; l(e) = l(e') \Longrightarrow IN_C(e) = IN_{C'}(e')$.

Two distinct events $e \in C$, $e' \in C'$, fulfilling $l(e) = l(e') \wedge {}^\bullet e = {}^\bullet e'$, are called *weak identical*.

Definition 9 (Reduced token flow unfolding). *Let (N, m_0), $N = (P, T, F, W)$, be a marked p/t-net. A* reduced token flow unfolding of (N, m_0) *is a generalized token flow unfolding $((PES, l), (x_C)_{C \in Con_{pre}})$ satisfying (IN), (OUT), (INIT),*

$$\textit{(MIN):} \; \forall C \in Con_{pre}, \forall e, e' \in C, e \prec_s e' : \; (\exists p \in P : \; x_C(e, e')(p) \geqslant 1)$$

and having no weak identical events e, e' satisfying $\{e, e'\} \notin Con$.

Similar as for token flow unfoldings, prefixes can be defined. Given a reduced token flow unfolding $\mu = ((PES, l), (x_C)_{C \in Con_{pre}})$, $PES = (E, Con, \prec)$, each non-empty left-closed subset $E' \subseteq E$ defines a reduced token flow unfolding $\mu' = ((PES', l'), (x'_C)_{C \in Con'_{pre}})$, $PES' = (E', Con', \prec')$ by $Con' = \{C \in Con \mid C \subseteq E'\}$, $\prec' = \prec |_{E' \times E'}$, $l' = l|_{E'}$ and $x'_C = x_C$. Each token flow unfolding $\mu'' = ((PES'', l'), x')$, $PES'' = (E', Con'', \prec')$, $Con'' \subseteq Con'$ is called prefix of μ, denoted by $\mu'' \sqsubseteq \mu$.

Events can be appended to reduced token flow unfoldings similar as to token flow unfoldings. The residual token flow $Res_C(e)$ and the residual marking $Mar(C)$ are defined analogously as before, using x_C instead of x. If there are enough tokens in the residual marking to fire a transition t, in general a new token flow distribution for the considered consistency set has to be stored in order to have the possibility to append a respective transition occurrence to the consistency set via an enabling function y. Formally, such a token flow redistribution is given by a redistribution flow function $x : \prec |_{C \times C} \to \mathbb{N}^P$ fulfilling (IN), (OUT), (INIT) and $\forall (e, e') \in \prec_s \cap\, C \times C : (\exists p \in P : x(e, e')(p) > 0)$ such that there is an enabling function $y : C \to \mathbb{N}^P$ satisfying $\forall e \in C : \; W(l(e), p) - \sum_{e \prec e'} x(e, e')(p) \geqslant y(e)$ and $\forall p \in P : \; (\sum_{e \in C} y(e))(p) = W((p, t))$. The functions x and y define a new event $e_{x,y}$ through $l'(e_{x,y}) = t$, ${}^\bullet e_{x,y} =$

$\{e \in C \mid \exists e' : y(e') \neq 0 \wedge (e = e' \vee e \prec e')\}$ and $\forall e \in {}^\bullet e_{x,y} : x'(e, e_{x,y}) = y(e)$. If there is already a weak identical event not belonging to the consistency set, then this weak identical event is added to the consistency set. Otherwise, the new event is added.

Definition 10 (Appending events). *Let (N, m_0), $N = (P, T, F, W)$, be a marked p/t-net and let $\mu = ((PES, l), (x_C)_{C \in Con_{pre}})$, $PES = (E, Con, \prec)$, be a reduced token flow unfolding of (N, m_0).*

Let $t \in T$ and $C \in Con_{pre}$, such that $Mar(C)(p) \geqslant W((p,t))$ for each p. Let x be a redistribution function with associated enabling function y and $e_{x,y}$ be the corresponding new event.

If there is no event $e \notin C$ which is weak identical to $e_{x,y}$, then we define a generalized prime token flow event structure $Ext(\mu, C, x, y, t) = ((PES', l'), x')$, $PES' = (E', Con', \prec')$, through $E' = E \cup \{e_{x,y}\}$, $l'|_E = l$, $Con' = Con \cup \{C' \cup \{e_{x,y}\} \mid C' \subseteq C\}$, $\forall C' \in Con'_{pre}$, $e_{x,y} \in C' : x'_{C'}|_\prec = x \wedge x'_{C'}|_{C' \times \{e_{x,y}\}} = x'$ and $\prec' |_{E \times E} = \prec$. We say that μ is extended by *$e_{x,y}$.*

If there is an event $e_{id} \notin C$ which is weak identical to $e_{x,y}$, then define a prime token flow event structure $Ext(\mu, C, x, y, t) = ((PES', l'), x')$, $PES' = (E', Con', \prec')$, updating *$Con$* by *$e_{new}$ through $E' = E$, $l' = l$, $\prec' = \prec$, $Con' = Con \cup \{C' \cup \{e_{id}\} \mid C' \subseteq C\}$ and $\forall C' \in Con'_{pre} \setminus Con_{pre}$, $e_{id} \in C' : x'_{C'}|_\prec = x \wedge x'_{C'}|_{C' \times \{e_{id}\}} = x'$. We say the μ* is updated by *$e_{x,y}$.*

In the reduced token flow unfolding, only one event having a certain set of pre-events is introduced (except for concurrent events in one run), although there are different possible distributions of the token flows on ingoing edges of the event. Only one example distribution of these possible token flows is stored.

For the reduced token flow unfolding analogous results hold as for token flow unfoldings. By construction $Ext(\mu, C, x, y, t)$ is a reduced token flow unfolding. Similar as for token flow unfoldings, a prefix relation $\sqsubseteq$ between reduced token flow unfoldings can be defined. Since through appending events, the token flow on old consistency sets is not changed, $\mu \sqsubseteq Ext(\mu, C, x, y, t)$ holds. Moreover, every finite reduced token flow unfolding μ can be constructed by a sequence of appending operations from $\mu_0 = (((\{e_{init}\}, \{\{e_{init}\}\}), \emptyset), id), \emptyset)$, where C, x, y and t are chosen according to μ.

Appending events to a token flow unfolding in different orders leads to reduced token flow unfoldings with isomorphic underlying prime event structures. Isomorphic prefixes (in different such reduced token flow unfoldings) may have different token flow distributions, representing processes with isomorphic underlying runs. In this sense, the order of appending events plays no role and we can define a maximal (w.r.t. $\sqsubseteq$) reduced token flow unfolding $Unf_{red}(N, m_0)$ as the limit of a sequence of finite token flow unfoldings $(\mu_n)_{n \in \mathbb{N}}$ with $\mu_{n+1} = Ext(\mu_n, C, x, y, t)$. $Unf_{red}(N, m_0)$ is unique up to isomorphism of the underlying prime event structure and up to the token flow stored for a consistency set, where different possible token flows produce isomorphic runs.

Each left-closed consistency set C of $Unf_{red}(N, m_0)$ represents a process π_C of (N, m_0) in the sense that the LPO $\mathrm{lpo}_C = (C, \prec |_{C \times C}, l|_C)$ is the run underlying π_C and x_C is the canonical token flow function of π_C. Moreover, in $Unf_{red}(N, m_0)$ all runs underlying a process of (N, m_0) are represented by consistency sets (without loss of generality we can start the construction of $Unf_{red}(N, m_0)$ with an arbitrary process representing a specific run).

$Unf_{red}(N, m_0)$ avoids the generation of processes with isomorphic underlying runs. Namely, only in special auto-concurrency situations runs of (N, m_0) are represented more than once in $Unf_{red}(N, m_0)$. It can be seen similar as for token flow unfoldings that, if there are two sets $C \neq C' \in Con_{pre}$ representing processes with isomorphic underlying runs, then there exist events $e \in C, e' \in C', e \neq e'$ such that e and e' are weak identical and fulfill $\{e, e'\} \in Con$. That means, if (N, m_0) allows no auto-concurrency, there is a one-to-one correspondence between left-closed consistency sets of $Unf_{red}(N, m_0)$ and (isomorphism classes of) runs.

As a topic of future research, similar as for strong identical events and isomorphic processes, to avoid isomorphic runs, we can generalize the notion of weak identical events in the sense that two weak identical events have not necessarily common, but weak identical pre-events. This leads to the notion of *isomorphic weak identical* events analogously as for isomorphic strong identical events.

4 Algorithms

In this section we briefly describe two algorithms to construct unfolding models of a marked p/t-net with finite behavior. The algorithms essentially follow the Definitions 8 and 10. We implemented both methods.

The first algorithm computes a token flow unfolding equal to the maximal token flow unfolding, except that some isomorphic processes caused by auto-concurrency of transitions are omitted. Starting with the token flow unfolding consisting only of the initial event e_{init} and having the only consistency set $\{e_{init}\}$, events are appended to maximal left-closed consistency sets in a breadth-first way. In each iteration step, the algorithm picks the next consistency set C and, for each transition $t \in T$, stores all enabling functions for appending a t-occurrence. The enabling functions can be computed from the residual token flows of events $e \in C$ and the residual marking of C. Finding all possible choices of enabling functions is a combinatorial problem. For each enabling function, a new event is generated and the old token flow unfolding is either extended or updated by the new event in a similar way as described in Definition 8. In contrast to Definition 8, in each appending step only maximal left-closed consistency sets are constructed. Therefore, in some special cases of auto-concurrency the described algorithm does not construct all possible isomorphic strong identical events. That is because not all left-closures of subsets of strong identical events are considered as consistency sets (if there are two concurrent strong identical events, one is appended first and the second is only appended to consistency sets including the first appended event). Therefore, in general the calculated token flow unfolding contains less events than the maximal token flow unfolding. But calculating these events (which are isomorphic strong identical to already appended events) would only lead to isomorphic processes (i.e. the unfolding computed by the algorithm still represents all processes) and would worsen the runtime.

The second algorithm computes a reduced token flow unfolding equal to the maximal reduced token flow unfolding, except that only processes with minimal causality are represented. Starting with the token flow unfolding consisting only of the initial event e_{init} and having the only consistency set $\{e_{init}\}$, the algorithm essentially appends events to prefixes of left-closed consistency sets in a breadth-first way. In each

iteration step, the algorithm picks the next consistency set C and tries to append each transition t to prefixes of C. The aim is to find all minimal prefixes which allow to append a t-occurrence. We say that a transition occurrence can be appended to a prefix of a consistency set C, if there exists a token flow function fulfilling (IN), (OUT) and (INIT) of the resulting LPO. In [11] a polynomial algorithm to check this and to construct such a token flow function in the positive case was presented. For each computed token flow function, a new event is generated and the old token flow unfolding is either extended or updated by the new event in a similar way as described in Definition 8 (the computed token flow function defines the redistribution function and the enabling function). Since the algorithm appends transition occurrences only to minimal prefixes of C for which this is possible, the resulting reduced token flow unfolding contains all runs with minimal causality of the given p/t-net. In this sense it represents the complete partial order behavior. Observe that weak identical events are only constructed in cases of auto-concurrency, and that only left-closed consistency sets are constructed (each appending step leads a maximal left-closed consistency set). In contrast to the construction algorithm of the token flow unfolding, here all isomorphic runs appearing through auto-concurrency of events are computed, because all left-closures of subsets of weak identical events are considered as prefixes of consistency sets.

5 Experimental Results

In this section we experimentally test our implementation of the construction algorithms having the standard unfolding algorithm as a benchmark.

				Standard unfolding				Token flow unfolding				Reduced token flow unfolding			
n	m	x	y	E	P	time	mem	E	P	time	mem	E	P	time	mem
1	3	2	3	19	12	82ms	1133kb	5	2	25ms	557kb	4	1	42ms	625kb
2	4	2	3	267	132	1406ms	2548kb	15	6	43ms	667kb	9	4	76ms	865kb
1	3	4	2	91	315	2721ms	2365kb	11	3	37ms	704kb	7	1	59ms	917kb
1	3	4	3	175	840	23622ms	4640kb	9	6	40ms	685kb	5	1	55ms	754kb
3	4	2	3	799	612	90665ms	5067kb	22	10	54ms	761kb	12	7	103ms	1160kb
3	4	4	3	-	-	-	-	43	56	271ms	2440kb	13	3	102ms	1159kb
3	4	5	3	-	-	-	-	45	104	672ms	6196kb	17	7	178ms	1149kb
	n	m	k												
	1	1	1	41	22	133ms	1011kb	13	6	47ms	834kb	13	6	103ms	1135kb
	1	2	1	47	28	180ms	1270kb	13	6	49ms	841kb	13	6	114ms	1185kb
	2	1	1	71	58	469ms	1325kb	17	9	65ms	917kb	15	7	145ms	1211kb
	2	2	1	77	67	694ms	1438kb	17	9	65ms	917kb	15	7	145ms	1235kb
	1	1	2	-	-	-	-	179	150	640ms	5200kb	71	68	8561ms	2580kb
	2	2	2	-	-	-	-	239	413	3498ms	16991kb	95	147	54371ms	5414kb

Fig. 7. Experimental results: E shows the number of events and P the number of maximal processes in the constructed unfolding.

To construct the standard unfolding, we use an adapted version of the unfolding algorithm in [4]. When interpreting the results, one has to pay attention that this unfolding algorithm is not completely runtime optimized, but the remaining improvement

potential should be limited. We compare the runtime, memory consumption as well as the size and the number of maximal processes of the resulting event structures. The upper table in Figure 7 shows a test of the parameterized version of the example net of Figure 1 shown in Figure 8. The lower table in Figure 7 shows a test of the net in Figure 9 modeling for example a coffee automata.

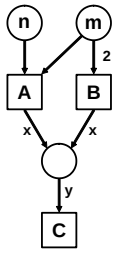

Fig. 8. Parameterized test net N_1.

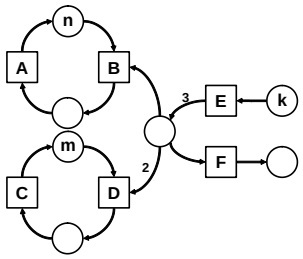

Fig. 9. Parameterized test net N_2.

The experimental results indicate that our new unfolding approaches are superior to the standard approach. For the tested examples, the runtime, memory consumption and the sizes of the resulting structure of our new algorithms are a lot better. It is clear that the standard unfolding is least as big as the token flow unfolding and the reduced token flow unfolding, but usually considerably bigger, if the net contains arc weights or a non-safe initial marking. In these cases our new algorithms are significantly faster and use less memory. Comparing the two new approaches shows that in almost every tested case the calculated reduced token flow unfolding is actually smaller than the calculated token flow unfolding, but the redistribution of token flows in each step worsens the runtime.

6 Conclusion

In this paper we propose two new unfolding semantics for p/t-nets based on the concepts of prime event structures and token flows. The definitions of the two unfolding models are motivated by algorithmic aspects. We develop a construction algorithm for both unfolding models, if they are finite. We show that there are many cases in which our implemented algorithms are significantly more efficient than standard unfolding methods for p/t-nets.

We finally want to mention that the two presented unfolding models are a conservative extension of the unfolding model introduced in [5] for safe nets. That means, for safe nets, the standard unfolding, the token flow unfolding and the reduced token flow unfolding coincide.

Topic of further research is the application of isomorphic weak resp. strong identical events to avoid isomorphic runs resp. isomorphic processes at all, the adaption of the theory of complete finite prefixes to our approach and the adaption of model checking algorithms. Although there are complete finite prefixes which also avoid redundant events, we believe that our approach yields faster construction algorithms since such complete finite prefixes rely on complex adequate orders which cannot be implemented efficiently.

References

1. E. Best and R. Devillers. Sequential and concurrent behaviour in petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.
2. J.-M. Couvreur, D. Poitrenaud, and P. Weil. Unfoldings for general petri nets. *http://www.labri.fr/perso/weil/publications/depliage.pdf*, University de Bordeaux I (Talence, France), University Pierre et Marie Curie (Paris, France), 2004.
3. J. Desel, G., and C. Neumair. Finite unfoldings of unbounded petri nets. In J. Cortadella and W. Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2004.
4. J. Desel, G. Juhás, and R. Lorenz. Viptool-homepage., 2003. http://www.informatik.ku-eichstaett.de/projekte/vip/.
5. J. Engelfriet. Branching processes of petri nets. *Acta Informatica*, 28(6):575–591, 1991.
6. J. Esparza and K. Heljanko. Implementing ltl model checking with net unfoldings. In M. B. Dwyer, editor, *SPIN*, volume 2057 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 2001.
7. J. Esparza, S. Römer, and W. Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
8. U. Goltz and W. Reisig. The non-sequential behaviour of petri nets. *Information and Control*, 57(2/3):125–147, 1983.
9. S. Haar. Branching processes of general s/t-systems and their properties. *Electr. Notes Theor. Comput. Sci.*, 18, 1998.
10. P. Hoogers, H. Kleijn, and P. Thiagarajan. An event structure semantics for general petri nets. *Theoretical Computer Science*, 153(1&2):129–170, 1996.
11. G. Juhás, R. Lorenz, and J. Desel. Can i execute my scenario in your net?. In G. Ciardo and P. Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 289–308. Springer, 2005.
12. V. Khomenko, A. Kondratyev, M. Koutny, and W. Vogler. Merged processes: a new condensed representation of petri net behaviour. *Acta Inf.*, 43(5):307–330, 2006.
13. V. Khomenko and M. Koutny. Towards an efficient algorithm for unfolding petri nets. In K. G. Larsen and M. Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2001.
14. V. Khomenko and M. Koutny. Branching processes of high-level petri nets. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 458–472. Springer, 2003.
15. V. Khomenko, M. Koutny, and W. Vogler. Canonical prefixes of petri net unfoldings. *Acta Inf.*, 40(2):95–118, 2003.
16. K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In G. von Bochmann; D. K. Probst, editor, *CAV*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 1992.
17. J. Meseguer, U. Montanari, and V. Sassone. On the model of computation of place/transition petri nets. In R. Valette, editor, *Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 16–38. Springer, 1994.
18. J. Meseguer, U. Montanari, and V. Sassone. On the semantics of place/transition petri nets. *Mathematical Structures in Computer Science*, 7(4):359–397, 1997.
19. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part i. *Theoretical Computer Science*, 13:85–108, 1981.
20. G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.