Hasse Diagram Generators and Petri Nets

Mateus de Oliveira Oliveira
School of Computer Science, Tel Aviv University, Tel Aviv, Israel
mateusde@tau.ac.il

Blavatnik School of Computer Science, Tel-Aviv University, Ramat Aviv, Tel-Aviv 69978, Israel

Abstract. In [18] Lorenz and Juhás raised the question of whether there exists a suitable formalism for the representation of infinite families of partial orders generated by Petri nets. Restricting ourselves to bounded p/t-nets, we propose *Hasse diagram generators* as an answer. We show that Hasse diagram generators are expressive enough to represent the partial order language of any bounded p/t net. We prove as well that it is decidable both whether the (possibly infinite) family of partial orders represented by a given Hasse diagram generator is included in the partial order language of a given p/t-net and whether their intersection is empty. Based on this decidability result, we prove that the partial order languages of two given Petri nets can be effectively compared with respect to inclusion. Finally we address the synthesis of k-safe p/t-nets from Hasse diagram generators.

Key words: Causality/partial order theory of concurrency

1 Introduction

Partial orders provide an intuitive way to formalize concurrent system scenarios. In particular, whenever it is possible to model a concurrent system by means of a p/t-net, we may regard scenarios as the partial orders induced by the events of p/t-net processes [14, 7] . These partial orders may represent one out of two possible semantics: The causal and the execution semantics. Within the causal semantics, an event v is connected to an event v', i.e. $(v < v')$, if and only if v' causally depends on the occurrence of v. The absence of an edge between two events indicates independence. The execution semantics is not aimed to indicate a causal dependence between events, but rather the order in which they are executed. An edge connecting v to v' indicates that v' should not occur before v. The absence of an edge between v and v' indicates that no restriction is imposed on the order in which they are executed. We say that a partial order po is a *causal order* of a given p/t-net N if it is associated to N with the causal semantics and an *execution* if it is associated to N with the execution semantics. We denote $\mathcal{L}_{cau}(N)$ the set of causal orders of N, and $\mathcal{L}_{ex}(N)$ its set of executions.

With the aim of finitely representing the whole behavior of systems, i.e., possibly infinite families of partial orders, we introduce *Hasse Diagram Generators*. This formalism turns to be expressive enough to allow the specification of the behavior of any bounded p/t-net. More precisely, we prove in theorem 7.8 that for each bounded p/t-net N, there exist effectively computable Hasse diagram generators $\mathcal{HG}_{ex}(N)$ and $\mathcal{HG}_{cau}(N)$ whose partial order languages match respectively the set of executions of N and the set of causal orders of N. This result settles a question posed in [18] for the case of bounded p/t-nets.

It is often useful to verify whether the behavior of a given system subsumes a collection of desired scenarios or whether it contains some out of a collection of undesired scenarios. Our main result concerns the decidability of these verification tasks whenever the systems are modeled by bounded p/t-nets and the scenarios specified by Hasse diagram generators. Below we use the

variable *sem* to indicate that a result is valid for both the causal semantics ($sem = cau$) and the execution semantics ($sem = ex$). Within this convention our *verification theorem* (Theorem 7.3) states that for any Hasse diagram generator $\mathcal{HG}$, and any bounded p/t-net N, it is decidable whether the language $\mathcal{L}_{PO}(\mathcal{HG})$ of partial orders generated by $\mathcal{HG}$ is included in $\mathcal{L}_{sem}(N)$, as well as whether the intersection of $\mathcal{L}_{PO}(\mathcal{HG})$ with $\mathcal{L}_{sem}(N)$ is empty. Previously, decidability was stated for single (and consequently finite families of) partial orders carrying both semantics [17].

Comparison of the behavior of two given p/t-nets N_1 and N_2 follows as a corollary (corollary 7.9) of theorems 7.3 and 7.8. Namely, suppose we wish to know whether each scenario of N_1 is a scenario of N_2, i.e, whether $\mathcal{L}_{sem}(N_1) \subseteq \mathcal{L}_{sem}(N_2)$. By using our expressibility result (Theorem 7.8) we compute the Hasse diagram generator $\mathcal{HG}_{sem}(N_1)$ which specifies precisely the behavior of N_1. Then using our verification result (Theorem 7.3), we verify whether $\mathcal{L}_{PO}(\mathcal{HG}_{sem}(N_1)) \subseteq \mathcal{L}_{sem}(N_2)$. We highlight however that comparison of the partial order languages of two arbitrary Hasse diagram generators is undecidable (Theorem 2.6). We observe that for the comparison of the behaviors of nets, we do not need to compute the Hasse diagram generator corresponding to the second net N_2. This result also implies that the equivalence of partial order behaviors of bounded p/t-nets is decidable. Previously decidability had been established only for the restricted case of 1-safe nets [16, 13].

We apply our main results to the study of the synthesis of p/t-nets from Hasse diagram generators. More precisely, given a Hasse diagram generator $\mathcal{HG}$ and $k \in \mathbb{N}$ we address the problem of synthesizing a k-safe [1] p/t-net whose partial order behavior is minimal[2] with respect to inclusion of the behavior specified by $\mathcal{HG}$. Our approach is based on the theory of regions. This theory was founded by Ehrenfeucht and Rozenberg in [11] in the context of partial 2-structures. It was used by the same authors in [12] to characterize partial 2-structures that are isomorphic to the reachability graph of elementary net systems. Subsequently the theory of regions was generalized and applied to the synthesis of several types of Petri nets, from various types of automata and languages, specifying sequential and step behaviors [15, 1, 2, 8, 9]. In [18] an abstract notion of region was defined for partial order languages carrying the execution semantics. This notion was effected in [4, 6] for finite sets of partial orders and in [5] for a term based representation of partial orders, which is not expressive enough to represent the partial order behavior of arbitrary k-safe p/t-nets. In the present work we develop the notion of k-safe region for Hasse diagram generators and employ it to derive a k-safe p/t-net whose set of executions minimally includes the behavior specified by $\mathcal{HG}$. Imposing an additional bound r on the number of repeated copies of each place on the synthesized net we obtain a similar result with respect to its set of causal orders. We notice that the present work addresses for the first time the synthesis of (unlabeled) p/t-nets from partial order languages carrying the causal semantics.

The central technical contribution of our paper, which is a pre-requisite for the results listed above is a characterization of Hasse diagrams of p/t-net executions and causal orders in terms of what we call *p-interlaced flows*. These interlaced flows may be regarded as a refinement of the token flows defined in [17, 3] for partial orders. The crucial difference being that token flows can not be coherently defined only on the edges belonging to the Hasse diagram of a partial order, while interlaced flows do. As a consequence, interlaced flows may be transposed to Hasse diagram generators in order to characterize whole families of p/t-net causal orders and executions.

[1] We use the term k-safe to indicate that the bound k must be given as a parameter for the synthesis, as opposed to the term bounded, which implies the k-safeness for some a priori unknown k.

[2] Among all other k-safe p/t-nets.

We regard directed acyclic graphs *DAGs* as being the result of the composition of smaller pieces, which we call *slices*. Essentially Hasse diagram generators are automata that assemble these slices in such a way that all DAGs they generate are Hasse diagrams. In order to assemble these small pieces coherently, we will need to provide a "sliced" characterization of Hasse diagrams. This will be done in in Theorem 6.2.*iv* by using the concept of *seasoner*. Seasoners will be used as well to provide sliced characterizations of Hasse diagrams of p/t-nets executions and causal orders.

The rest of the paper is organized as follows: next, in Section 2 we introduce slices and Hasse diagram generators. Subsequently, in Section 3 we define p/t-nets and their processes, from which executions and causal orders are derived. Section 4 highlights some of the differences between causal and execution semantics of partial order specifications. In Section 5 we develop the notion of interlaced flow as a pre-requisite for the statement of our main results. Section 6 discusses *seasoners* which are used to provide local characterizations of Hasse diagrams and of p/t-net scenarios. In Section 7 we state our main results, namely, the verification, expressibility and net comparison theorems. In section 8 we deal with the synthesis of k-safe p/t-nets from Hasse diagram generators. We end our paper by making some final comments in Section 9.

2 Slices, slice graphs and Hasse diagram generators

In this section we introduce the notions of *slice* and *slice languages*, from which infinite languages of partial orders are derived. We provide concrete representations of slice languages by means of *slice graphs*. *Hasse diagram generators* are then introduced as a subclass of slice graphs. We finish the section with a proof that both the inclusion and the emptiness of intersection of partial order languages associated to Hasse diagram generators are in general undecidable.

Graphs and DAGs A *labeled directed multi-graph*, which for simplicity we will refer to as *graph*, is a triple $G = (V, E, l)$ where V is a set of *vertices*, $l : V \to \mathcal{X}$ is a function that labels the vertices in V with elements of an arbitrary set of labels $\mathcal{X}$, and $E \subset \mathbb{N}^{V \times V}$ is a multiset of *edges* (ordered pairs of vertices). In the sequel, for an edge $e = (v_1, v_2) \in E$ we will often write e^s and e^t to respectively denote the source (v_1) and target (v_2) of e. A path in a graph is an alternated sequence of vertices and edges $v_1 e_1 v_2 ... v_k e_k v_{k+1}$ such that $e_i^t = v_{i+1} = e_{i+1}^s$. A cycle is a path $v_1 e_1 v_2 ... v_k e_k v_{k+1}$ with $v_1 = v_{k+1}$. A loop is a cycle with a unique edge, i.e. e such that $e^s = e^t$. A *directed acyclic graph* (DAG) is a graph without cycles, but possibly containing multiple edges. We say that a vertex of a DAG is *minimal* (*maximal*) if it is not the target (source) of any edge.

Slice: A *slice* is a DAG $\mathbf{S} = (V, E, l)$ where $V = I \dot{\cup} C \dot{\cup} O$ and $l : V \to T \cup \mathbb{N}$. The vertex set V is partitioned into three subsets: A non-empty center C labeled by l with elements of T [3] and the in- and out-frontiers I and O respectively which are numbered by l in such a way that $l(I) = [|I|]$ and $l(O) = [|O|]$ where $[n]$ denotes the set $\{1, ..., n\}$. Furthermore a unique edge touches each frontier vertex $v \in I \dot{\cup} O$. This edge is outgoing if v lies on the in-frontier I and incoming if v lies on the out-frontier O.

In drawings, we surround slices by dashed rectangles, and implicitly direct their edges from left to right. In-frontier and out-frontier vertices are determined respectively by the intersection of edges with the left and right sides of the rectangle. Frontier vertices are implicitly numbered from up to down. Center vertices are indicated by their labels (Fig. 1-*i*).

[3] Here T is just a set of symbols. Later we will regard T as being the set of transitions of some p/t-net.

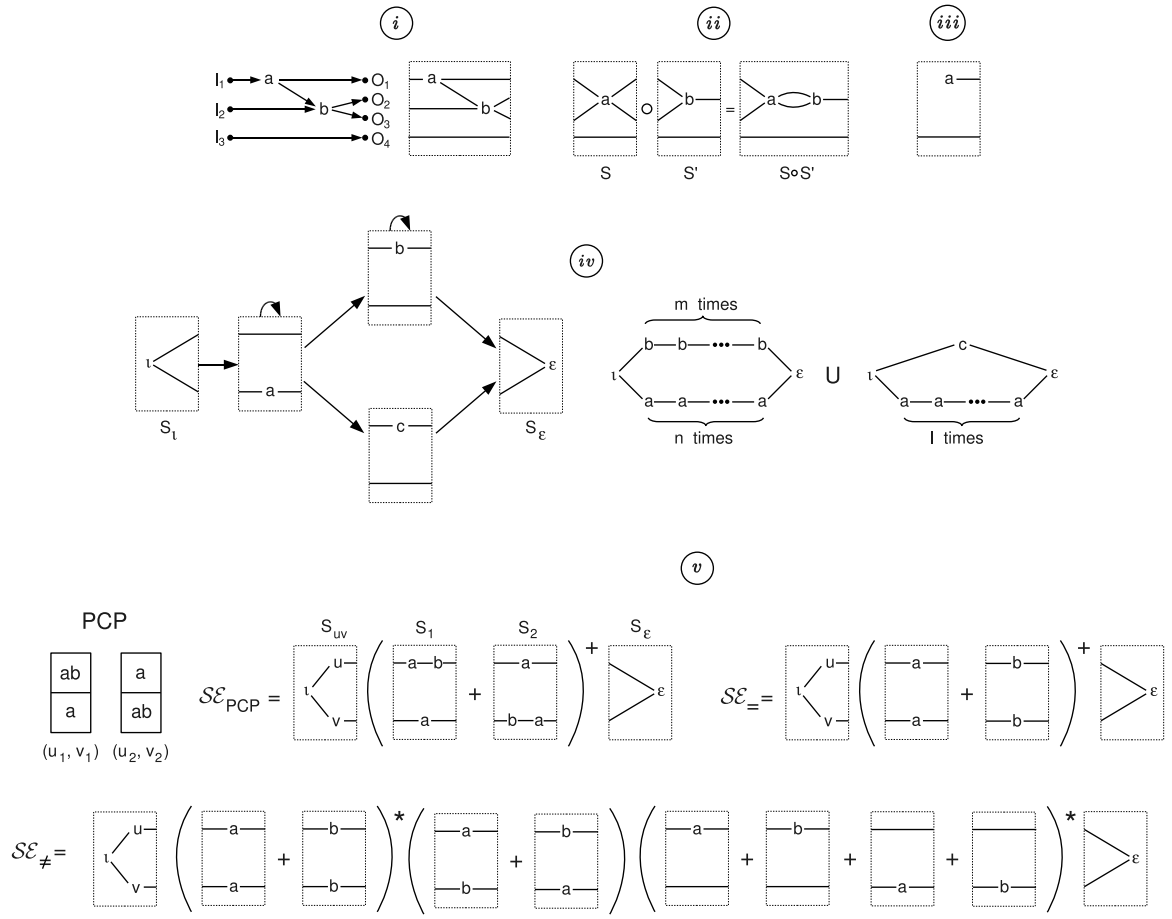

Fig. 1. *i*) A slice and its representation according to our convention. *ii*) Composition of slices. *iii*) A slice which is not standard. *iv*) A slice graph labeled with unit slices, and an intuitive representation of its graph language. $\mathbf{S}_\iota$ is initial and $\mathbf{S}_\varepsilon$, final. *v*) reduction of the Post correspondence problem to the intersection and to the inclusion of graph languages. Obs: We write $(\cdots)^+$ in place of $(\cdots)(\cdots)^*$.

Initial, Final, Unit and Standard Slices We say that a slice is *initial* if its in-frontier is empty and *final* if its out-frontier is empty. A slice with a unique vertex in the center is called a *unit slice*. In this paper we assume that unit slices cannot be initial and final at the same time. Unit initial slices have their center vertex labeled with ι and unit final slices have their center vertex labeled with ε. We assume that ι and ε never label a center vertex that is connected to both the in and out frontiers of a slice. A unit slice $\mathbf{S}$ is *standard* if there is at least one edge connecting its center vertex to an in(out)-frontier vertex whenever the in(out)-frontier of $\mathbf{S}$ is not empty. In figure 1.*iii* we depict a unit slice that is not standard. All the other unit slices in figure 1 are standard.

Composition of Slices: Let $\mathbf{S}_1 = (V_1, E_1, l_1)$ and $\mathbf{S}_2 = (V_2, E_2, l_2)$ be two distinct slices where $V_1 = I_1 \dot\cup C_1 \dot\cup O_1$ and $V_2 = I_2 \dot\cup C_2 \dot\cup O_2$ are disjoint sets of vertices. We say that $\mathbf{S}_1$ can be composed with $\mathbf{S}_2$ whenever the out-frontier of $\mathbf{S}_1$ is of the same size as the in-frontier of $\mathbf{S}_2$, i.e., $|O_1| = |I_2|$. In this case, the resulting slice $\mathbf{S}_1 \circ \mathbf{S}_2$ is obtained by gluing the single edge touching the j-th out-frontier vertex of $\mathbf{S}_1$ to the corresponding edge touching the j-th in-frontier vertex of $\mathbf{S}_2$ (Fig. 1-*ii*). We note that as a result of the composition, multiple edges may arise, since the vertices on the glued frontiers disappear. Formally we have

$$\mathbf{S}_1 \circ \mathbf{S}_2 = [(\mathbf{S}_1 - O_1)\dot{\cup}(\mathbf{S}_2 - I_2)] + \{(e_1^s, e_2^t) \mid l(e_1^t) = j = l(e_2^s), e_1 \in E_1, e_2 \in E_2, j \in \mathbb{N}\},$$

where $\dot{\cup}$ stands for the disjoint union of multigraphs. The minus operation stands for the usual deletion of vertices and the + operation for the usual addition of edges in multigraphs. It is easy to see that any DAG, even containing multiple edges, can be cast as the composition of a sequence of unit slices. In particular, if the DAG has unique minimal and maximal vertices, then it can be cast as the composition of a sequence of standard slices.

Definition 2.1 (Slice Language). *Let $\Sigma_{\mathbb{S}}$ be a slice alphabet, i.e, a set of slices which are regarded as symbols. We say a regular language $\mathcal{L} \subseteq \Sigma_{\mathbb{S}}^*$ is a* slice language *if for every word $\mathbf{S}_1\mathbf{S}_2 \cdots \mathbf{S}_n \in \mathcal{L}$, $\mathbf{S}_1$ is initial, $\mathbf{S}_n$ final and $\mathbf{S}_i$ can be composed with $\mathbf{S}_{i+1}$ for $1 \leq i \leq n-1$.*

It will be convenient for the purposes of this paper, to represent slice languages through slice graphs defined below, instead of using regular expressions or finite automata over slice alphabets. An exception will be made in Theorem 2.6 whose proof involves regular expressions over slices. Before we state a proposition relating labeled walks on graphs and regular languages.

Proposition 2.2 (Labeled Walks and Regular Languages). *Let Σ be an alphabet. A language $\mathcal{L} \subseteq \Sigma^+$ is regular if and only if there exists a graph $G = (V, E, l)$ with $l : V \to \Sigma$, a set of initial vertices $V_\iota \subseteq V$ and a set of final vertices $V_\varepsilon \subseteq V$ such that*

$$\mathcal{L} = \mathcal{L}(G) = \{l(v_1)l(v_2)...l(v_n) : \ v_1v_2...v_n \text{ is a walk in } G \text{ from } v_1 \in V_\iota \text{ to } v_n \in V_\varepsilon\}$$

For the sake of completeness we include a proof of Proposition 2.2 in the appendix. Basically this proposition says that the set of labeled walks in a graph, from vertices marked as initial to vertices marked as final, forms a regular language. Conversely every regular language can be represented as such a set of labeled walks. Proposition 2.2 implies that the slice graphs we define next are indeed adequate to represent slice languages, in the sense that a language over a slice alphabet is a slice language if and only if it can be generated by a slice graph.

Definition 2.3 (Slice Graph). *A* slice graph *over a slice alphabet $\Sigma_{\mathbb{S}}$ is a directed graph $\mathcal{SG} = (\mathcal{V}, \mathcal{E}, \mathcal{S})$ possibly containing loops but free of multiple edges. The function $\mathcal{S} : \mathcal{V} \to \Sigma_{\mathbb{S}}$ satisfies the following condition: $(v_1, v_2) \in \mathcal{E}$ implies that $\mathcal{S}(v_1)$ can be composed with $\mathcal{S}(v_2)$. We say that a vertex on a slice graph is initial if it is labeled with an initial slice and final if it is labeled with a final slice. We call $\mathcal{L}(\mathcal{SG})$ the slice language generated by $\mathcal{SG}$, which we define as:*

$$\mathcal{L}(\mathcal{SG}) = \{\mathcal{S}(v_1)\mathcal{S}(v_2)...\mathcal{S}(v_n) : v_1v_2...v_n \text{ is a walk on } \mathcal{SG} \text{ from an initial to a final vertex}\}$$

Partial Orders and Hasse Diagrams: A graph G is *simple* if it does not contain multiple edges. The *simplification* $simpl(G)$ of a graph G is the maximal simple subgraph of G. The transitive closure of a simple graph $G = (V, E, l)$ is the graph $G^* = (V, E^*, l)$ where E^* is the transitive closure of E. The transitive reduction of a DAG $G = (V, E, l)$, or alternatively, the Hasse diagram induced by G, is the unique minimal simple subgraph $H = (V, E', l)$ of G satisfying $H^* = simpl(G)^*$. More generaly, we say that any simple DAG H that is equal to its transitive reduction is a Hasse diagram. A *partial order* is a simple DAG $po = (V, <, l)$ where $< \subset V \times V$ is irreflexive and transitive. The partial order induced by a DAG G is the transitive closure of the simplification of G.

Definition 2.4 (Languages derived from Slice Languages). *Let $\mathcal{L}$ be a slice language over $\Sigma_{\mathbb{S}}$. We derive from $\mathcal{L}$ the two following languages:*

(i) Graph language: $\mathcal{L}_G = \{\mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n | \mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}\}$
(ii) Partial order language: $\mathcal{L}_{PO} = \{H^* | H \in \mathcal{L}_G\}$

If $\mathcal{SG}$ is a slice graph, we denote by $\mathcal{L}_G(\mathcal{SG})$ and $\mathcal{L}_{PO}(\mathcal{SG})$ respectively the graph language and partial order language derived from $\mathcal{L}(\mathcal{SG})$. If $\mathcal{L}, \mathcal{L}'$ are two slice languages, the following chain of implications follows from Definition 2.4:

$$\mathcal{L} \subseteq \mathcal{L}' \Rightarrow \mathcal{L}_G \subseteq \mathcal{L}'_G \Rightarrow \mathcal{L}_{PO} \subseteq \mathcal{L}'_{PO} \tag{1}$$

It is easy to verify by counterexamples that the converse of none of these implications is valid for general slice languages. We say that a slice language $\mathcal{L}$ is a *Hasse slice language* if its graph language $\mathcal{L}_G$ is constituted uniquely of Hasse diagrams. In other words, each element of $\mathcal{L}_G$ is the Hasse diagram of an element of $\mathcal{L}_{PO}$. We will represent Hasse slice languages trough Hasse diagram generators.

Definition 2.5 (Hasse Diagram Generator). *A Hasse diagram generator is a slice graph $\mathcal{HG}$ for which all elements of $\mathcal{L}_G(\mathcal{HG})$ are Hasse diagrams.*

We point out that for two Hasse slice languages $\mathcal{L}$ and $\mathcal{L}'$, $\mathcal{L}_{PO} \subseteq \mathcal{L}'_{PO}$ implies $\mathcal{L}_G \subseteq \mathcal{L}'_G$ but still does not necessarily imply $\mathcal{L} \subseteq \mathcal{L}'$. We end this section with a proof that inclusion and emptiness of intersection of graph languages and partial order languages derived from slice languages is in general undecidable.

Theorem 2.6. *Let $\mathcal{L}$ and $\mathcal{L}'$ be two slice languages over $\Sigma_{\mathbb{S}}$. Then it is undecidable whether $\mathcal{L}_G \subseteq \mathcal{L}'_G$ ($\mathcal{L}_{PO} \subseteq \mathcal{L}'_{PO}$) as well as whether $\mathcal{L}_G \cap \mathcal{L}'_G = \emptyset$ ($\mathcal{L}_{PO} \cap \mathcal{L}'_{PO} = \emptyset$).*

Proof. We reduce the Post correspondence problem over a two letter alphabet, which is undecidable [19], to both the emptiness of intersection and the inclusion of graph languages. Let $PCP = (\Gamma; (u_1, v_1), (u_2, v_2)..., (u_n, v_n))$ be an instance of the Post correspondence problem, where $\Gamma = \{a, b\}$ and (u_i, v_i) are pairs of non empty words over Γ. The Post correspondence problem asks whether for some $k \in \mathbb{N}$ there exist integers $m_1, m_2, \cdots, m_k \in [n]$ such that $u_{m_1}u_{m_2}...u_{m_k} = v_{m_1}v_{m_2}...v_{m_k}$. Consider the regular expressions over slices $\mathcal{SE}_=$ and $\mathcal{SE}_{\neq}$ which are depicted in figure 1-*v* and let $\mathcal{L}_G(\mathcal{SE}_=)$ and $\mathcal{L}_G(\mathcal{SE}_{\neq})$ be their respective graph languages. We note that each graph in $\mathcal{L}_G(\mathcal{SE}_=)$ corresponds to a pair (u, v) of non-empty words $u, v \in \Gamma^*$ for which $u = v$. Similarly, a graph belongs to $\mathcal{L}_G(\mathcal{SE}_{\neq})$ iff it corresponds to a pair of words (u, v) which differ in at least one position. Consider as well the PCP-instance-dependent regular expression $\mathcal{SE}_{PCP} = \mathbf{S}_{uv}(\mathbf{S}_1 + \mathbf{S}_2 + \cdots + \mathbf{S}_n)^+\mathbf{S}_\varepsilon$ where $\mathbf{S}_i$ is a slice corresponding to the pair (u_i, v_i) as exemplified in figure 1-*v*. As it is usually done for regular expressions, we write $(\cdots)^+$ in place of $(\cdots)(\cdots)^*$. It should be clear that PCP has a solution iff $\mathcal{L}_G(\mathcal{SE}_=) \cap \mathcal{L}_G(\mathcal{SE}_{PCP}) \neq \emptyset$. Conversely, PCP has no solution iff $\mathcal{L}_G(\mathcal{SE}_{PCP}) \subseteq \mathcal{L}_G(\mathcal{SE}_{\neq})$. Thus both inclusion and emptiness of intersection of graph languages are undecidable. We note that the languages generated by any of the slice expressions defined in this proof are Hasse slice languages. Hence $\mathcal{L}_{PO} \cap \mathcal{L}'_{PO} = \emptyset$ and $\mathcal{L}_{PO} \subseteq \mathcal{L}'_{PO}$ are undecidable as well.

Since a language $\mathcal{L}$ over a slice alphabet $\Sigma_{\mathbb{S}}$ is a slice language if and only if it is the slice language generated by a slice graph, we have:

Corollary 2.7. *Let $\mathcal{HG}$ and $\mathcal{HG}'$ be two slice graphs over a slice alphabet $\Sigma_{\mathbb{S}}$. Then it is undecidable whether $\mathcal{L}_{PO}(\mathcal{HG}) \subseteq \mathcal{L}_{PO}(\mathcal{HG}')$ as well as whether $\mathcal{L}_{PO}(\mathcal{HG}) \cap \mathcal{L}_{PO}(\mathcal{HG}') = \emptyset$.*

3 p/t-Nets and their Partial Order Semantics

p/t-Net: Let T be a finite set of transitions. Then a place over T is a triple $p = (p_0, \check{p}, \hat{p})$ where p_0 denotes the initial number of tokens in p and $\check{p}, \hat{p} : T \to \mathbb{N}$ are functions which denote the number of tokens that a transition t respectively puts in and takes from p. A p/t-net over T is a pair $N = (P, T)$ where T is a set of transitions and P a finite multiset of places over T. We assume through this paper that for each transition $t \in T$, there exist places $p_1, p_2 \in P$ for which $\check{p}_1(t) > 0$ and $\hat{p}_2(t) > 0$. A marking of N is a function $m : P \to \mathbb{N}$. A transition t is enabled at marking m if $m(p) \geq \hat{p}(t)$ for each $p \in P$. The occurrence of an enabled transition at marking m gives rise to a new marking m' defined as $m'(p) = m(p) - \hat{p}(t) + \check{p}(t)$. The initial marking m_0 of N is given by $m_0(p) = p_0$ for each $p \in P$. A sequence of transitions $t_0 t_1 ... t_{n-1}$ is an occurrence sequence of N if there exists a sequence of markings $m_0 m_1 ... m_n$ such that t_i is enabled at m_i and if m_{i+1} is obtained by the firing of t_i at marking m_i. A marking m is legal if it is the result of the firing of an occurrence sequence of N. A place p of N is k-safe if $m(p) \leq k$ for each legal marking m of N. A net N is k-safe if each of its places is k-safe. N is bounded if it is k-safe for some k. The union of two p/t-nets $N_1 = (P_1, T)$ and $N_2 = (P_2, T)$ having a common set of transitions T is the p/t-net $N_1 \cup N_2 = (P_1 \dot{\cup} P_2, T)$. We consider that the multiplicity of a place p in $P_1 \dot{\cup} P_2$ is the sum of its multiplicities in P_1 and in P_2.

Definition 3.1 (Process). *A process of a p/t-net $N = (P, T)$ is a DAG $\pi = (B \dot{\cup} V, F, \rho)$ where the vertex set $B \dot{\cup} V$ is partitioned into a set of* conditions *B and a set of* events *V. $F \subseteq (B \times V) \cup (V \times B)$ and $\rho : (B \cup V) \to (P \cup T) \cup \{\iota, \epsilon\}$ are required to satisfy the following conditions.*

1. *π has a unique minimal vertex $v_\iota \in V$ and a unique maximal vertex $v_\varepsilon \in V$.*
2. *Conditions are unbranched: $\forall b \in B, |\{(b, v) \in F\}| = 1 = |\{(v, b) \in F\}|$.*
3. *Places label conditions and transitions label events. Minimal and maximal vertices have special labels.*

$$\rho(B) \subseteq P \qquad \rho(V \backslash \{v_\iota, v_\varepsilon\}) \subseteq T \qquad \rho(v_\iota) = \iota \qquad \rho(v_\varepsilon) = \varepsilon$$

4. *If ρ labels an event $v \in V \backslash \{v_\iota, v_\varepsilon\}$ with a transition $t \in T$ then*
   (a) for each $p \in P$, v has $\hat{p}(t)$ preconditions labeled by p : $|\{(b, v) \in F : \rho(b) = p\}| = \hat{p}(t)$.
   (b) for each $p \in P$, v has $\check{p}(t)$ postconditions labeled by p : $|\{(v, b) \in F : \rho(b) = p\}| = \check{p}(t)$.
5. *For each $p \in P$, v_ι has p_0 post-conditions labeled by p : $|\{(v_\iota, b) : \rho(b) = p\}| = p_0$.*

The only point our definition of process differs from the usual definition of p/t-net process [14] is the addition of a minimal event v_ι which is labeled with a letter $\iota \notin T$ and a maximal event v_ε which is labeled with a letter $\varepsilon \notin T$. We notice that item 3.1.2 implies that every condition which is not connected to an event $v \in V$ labeled by a transition $t \in T$, is necessarily connected to v_ε. Intuitively, ι loads the initial marking of N and ε empties the marking of N after the occurrence of all events of the process. We call attention to the fact that the number of conditions connected to v_ε varies according to the process. These minimal and maximal events will be useful to avoid the consideration of particular cases in Theorem 5.5.

Prefix and Sequentialization A prefix of a partial order $po = (V, <, l)$ is a partial order $po' = (V', <', l')$ where $V' \subseteq V$, $<'$ is the restriction of $<$ to $V' \times V'$, $l' = l|_{V'}$ and for every $v' \in V', v \in V$ if $v < v'$ then $v \in V'$. A sequentialization of po is a partial order $po' = (V, <', l)$ where $< \subseteq <'$.

The causal order of a process π is obtained from it by abstracting its conditions and by considering the partial order induced by its events. An execution is a sequentialization of a causal order.

Definition 3.2 (Causal Orders and Executions of p/t-net Processes). *The causal order of a process $\pi = (B \dot{\cup} V, F, \rho)$ of a p/t-net N is the partial order $po_\pi = (V, <, l)$ where $<= F^*|_{V \times V}$ and $l = \rho|_V$. An execution of π is a sequentialization of po_π.*

We denote $\mathcal{L}_{cau}(N)$ the set of all causal orders derived from processes of N and $\mathcal{L}_{ex}(N)$ the set of all its executions. When talking about a prefix of a causal order (execution) of a process, we take the additional care of adding to it a maximal vertex labeled with ε, and connecting to this vertex all the maximal vertices of the original prefix. The slightly modified prefix is itself a causal order (execution) of N. We notice that with this proviso, $\mathcal{L}_{cau}(N)$ is prefix closed and $\mathcal{L}_{ex}(N)$ is both prefix and sequentialization closed.

4 Causal vs Execution Semantics

Any prefix of a causal order or of an execution of a p/t-net N is itself a causal order or execution of N [4]. Additionally, any sequentialization of an execution of N is also an execution of N. Thus while the causal language of N, $\mathcal{L}_{cau}(N)$, is closed by prefix, the execution language of N, $\mathcal{L}_{ex}(N)$, is both closed by prefix and sequentialization [20][4].

Closure under sequentialization brings to the execution language of a p/t-net some robustness which is not enjoyed by its causal language. For example, $\mathcal{L}_{ex}(N)$ remains the same if we multiply all the quantities defining the token game of N by the same constant. Also adding a place to N can at most restrict its execution behavior. If the added place is a linear combination of places already in N, then $\mathcal{L}_{ex}(N)$ does not change at all. In particular $\mathcal{L}_{ex}(N)$ remains invariant under repetition of places [5].

Lemma 4.1 ([4], [20]). *Let $N = (P, T)$ be a p/t-net $p = (p_0, \check{p}, \hat{p})$ a place in P and $\sigma \in \mathbb{Q}^+$ where $\mathbb{Q}^+$ is the set of non negative rational numbers. Denote σp the triple $(\sigma p_0, \sigma \check{p}, \sigma \hat{p})$ and $\sigma N = (\{\sigma p | p \in P\}, T)$. Then*

1. *if σN is a p/t-net, then $\mathcal{L}_{ex}(N) = \mathcal{L}_{ex}(\sigma N)$.*
2. *Let p' be a place over T. Then $\mathcal{L}_{ex}(N \cup (p', T)) \subseteq \mathcal{L}_{ex}(N)$. If additionally $p' = \sum_{p \in P} \sigma_p p$ for $\sigma_p \in \mathbb{Q}^+$, then $\mathcal{L}_{ex}(N) = \mathcal{L}_{ex}(N \cup (p', T))$.*

None of these properties are true with respect to the causal language of a net, as we show in figure 2. Multiplying a net by a constant may introduce causal orders which were not present in the causal language of the original net. The same is true if we repeat places. As a consequence, when imposing the causal semantics to partial order languages, the synthesis of p/t-nets from partial order languages turns out to be harder than if we were adopting the execution semantics. Still, specification through the causal semantics may be more precise and desirable for some applications. In section section 8, we will approach the synthesis for both cases.

5 Interlaced Flows, Executions and Causal Orders

In this section we introduce the notion of *p-interlaced flow* in order to characterize Hasse diagrams of p/t-net causal orders and executions. We show that a graph H is the Hasse diagram of a partial order corresponding to a p/t-net N if and only if for each place p of N, there exists a p-interlaced flow f_p which associates four values to each edge of H. Establishing an interlacing relation between these values, we are able to recover from the flows p/t-net processes whose causal order is the partial order induced by H.

[4] See proviso at the end of section 3.
[5] P is a multiset.

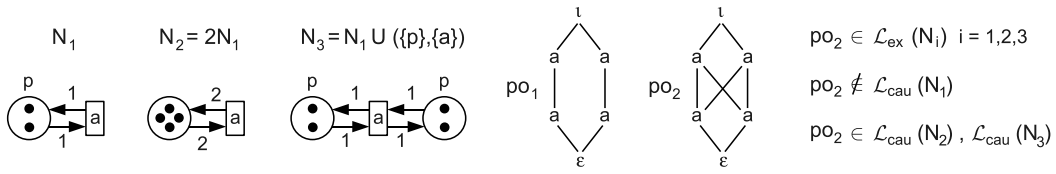

Fig. 2. In this figure we identify partial orders with their Hasse diagrams. po_1 is a causal order of N_1. po_2 is an execution of N_1, since it is a sequentialization of po_1. However po_2 is not a causal order of N_1. Multiplying N_1 by 2 or repeating the place p, we get the nets N_2 and N_3 which generate po_2 as a causal order.

Interlaced flows can be regarded as a refinement of token flows defined in [17, 3]. The main difference being that the latter needs to attach values to edges that do not belong to the transitive reduction of a partial order. As a consequence token flows are not adequate to provide local characterizations of p/t-nets causal orders and executions, which are a pre-requisite for the statement of our main results. In contrast in Section 6 we do provide such local characterizations by "slicing" interlaced flows.

Definition 5.1 (Interlaced Flow). *Let $H = (V, E, l)$ be a Hasse diagram. Then a four-tuple $f = (\boldsymbol{bb}, \boldsymbol{bf}, \boldsymbol{pb}, \boldsymbol{pf})$ of functions of type $E \to \mathbb{N}$ is called an interlaced flow in H if the following equation is satisfied for each vertex $v \in V$:*

$$\sum_{e^t=v} \boldsymbol{bf}(e) + \boldsymbol{pf}(e) = \sum_{e^s=v} \boldsymbol{pb}(e) + \boldsymbol{pf}(e) \tag{2}$$

Intuitively, for each $e \in E$, $\boldsymbol{pb}(e)$ counts some of the tokens produced in the **p**ast of e^s and consumed **b**y e^t; $\boldsymbol{pf}(e)$, some of the tokens produced in the **p**ast of e^s and consumed in the **f**uture of e^t, and $\boldsymbol{bf}(e)$, some of the tokens produced **b**y e^s and consumed in the future of e^t. Thus equation 2 states that on interlaced flows, the total number of tokens produced in the past of a vertex v, that arrives at it without being consumed, will eventually be consumed in the future of v. $\boldsymbol{bb}(e)$, which does not appear in equation 2 counts the total of tokens produced **by** e^s and consumed **by** e^t. This component of the flow will be used below in the definition of unit flows as well as in the definition of p-interlaced flows.

As we shall see in Lemma 5.3 any interlaced flow can be decomposed into a sum of unit flows. Each unit flow is aimed to keep track of the trajectory of a unique token, from the event where the token is created until the event where it is consumed. The sum of flows is performed componentwise. We write $f(e) = (\boldsymbol{bb}(e), \boldsymbol{bf}(e), \boldsymbol{pb}(e), \boldsymbol{pf}(e))$ for the 4-tuple of values associated by a flow f to an edge e of a Hasse diagram.

Definition 5.2 (Unit flow). *Let $H = (V, E, l)$ be a Hasse diagram and $w = v_1 e_1 v_2 ... v_k e_k v_{k+1}$ be a path in H. Then the* unit flow *of w in H, is the function $f_w : E \to \mathbb{N}^4$ where $f_w(e) = (0, 0, 0, 0)$ for $e \in E \backslash \{e_1, ..., e_k\}$ and $f_w(e_i)$ is defined as follows for $1 \leq i \leq k$:*

$$f_w(e_i) = \begin{cases} (1,0,0,0) & \text{if} \quad i = 1 \quad \text{and} \quad k = 1 \\ (0,1,0,0) & \text{if} \quad i = 1 \quad \text{and} \quad k > 1 \\ (0,0,0,1) & \text{if} \quad 1 < i < k \quad \text{and} \quad k > 1 \\ (0,0,1,0) & \text{if} \quad i = k \quad \text{and} \quad k > 1 \end{cases} \tag{3}$$

Intuitively f_w represents a unique token that is produced at the first vertex of w and is consumed at its last vertex, after traveling untouched along all intermediary vertices (fig 3.*iv*). At each edge, one component of the interlaced flow is equal to 1 and the other three are 0. If w has a unique edge, i.e. $k = 1$, the token produced by e_1^s is consumed by e_1^t, thus $\boldsymbol{bb}(e_1) = 1$.

For $k > 1$, the token produced by e_1^s travels to the future of e_1^t. In this case, $\boldsymbol{bf}(e_1) = 1$. For $1 < i < k$, the token is produced in the past of e_i^s and consumed in its future, hence $\boldsymbol{pf}(e_i) = 1$. Finally, for e_k we have $\boldsymbol{pb}(e_k) = 1$ since the token is produced in the past of e_k^s and consumed by e_k^t. It is easy to verify that for any path w of H, f_w is an interlaced flow in H. This and some other properties of interlaced flows are described in the following lemma, which will be used in the proof of Theorem 5.5.

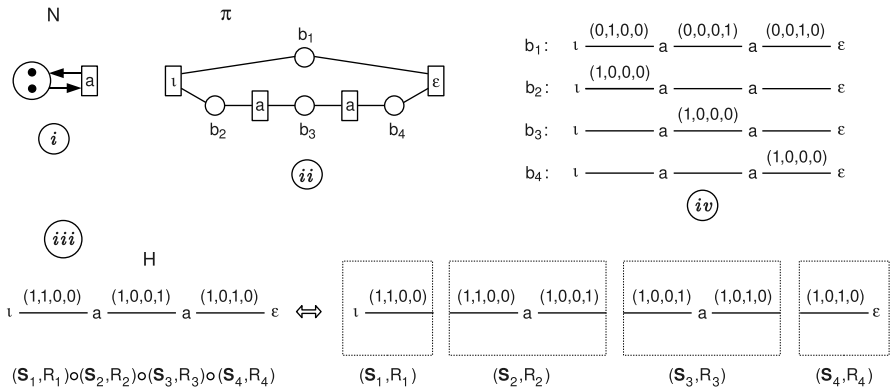

Fig. 3. *i*) A p/t-net N. *ii*) A process π of N. *iii*) The Hasse diagram H of the causal order derived from π together with an interlaced flow. H with the flow attached on it can be cast as the composition of a sequence of seasoned slices (Section 6). *iv*) The flow attached to H is decomposed into unit flows. One for each condition b_i of π.

Lemma 5.3. *Let $H = (V, E, l)$ be a Hasse diagram, w a path in H and $f, f' : E \to \mathbb{N}^4$ be two interlaced flows in H where $f = (\boldsymbol{bb}, \boldsymbol{bf}, \boldsymbol{pb}, \boldsymbol{pf})$ and $f' = (\boldsymbol{bb}', \boldsymbol{bf}', \boldsymbol{pb}', \boldsymbol{pf}')$. Then*

1. *Let $z : E \to \mathbb{N}^4$ be the zero flow, i.e., $z(e) = (0, 0, 0, 0)$ for every $e \in E$. Then z is an interlaced flow in H.*
2. *f_w is an interlaced flow in H.*
3. *$f + f'$ is an interlaced flow in H.*
4. *Let $w = v_1 e_1 v_2 ... v_k e_k v_{k+1}$ be a path in H and suppose $f = f' + f_w$, then*
   (a) $\boldsymbol{bb}(e_1) + \boldsymbol{bf}(e_1) = \boldsymbol{bb}'(e_1) + \boldsymbol{bf}'(e_1) + 1$
   (b) $\boldsymbol{pb}(e_k) + \boldsymbol{bb}(e_k) = \boldsymbol{pb}'(e_k) + \boldsymbol{bb}'(e_k) + 1$
5. *Let $e \in E$. Then*
   (a) $\boldsymbol{bb}(e) + \boldsymbol{bf}(e) \geq 1$ *if and only if there exists a path $w = v_1 e_1 v_2 .. v_k e_k v_{k+1}$ and an interlaced flow f' in H such that $e_1 = e$ and $f = f' + f_w$.*
   (b) $\boldsymbol{pb}(e) + \boldsymbol{bb}(e) \geq 1$ *if and only if there exists a path $w = v_1 e_1 v_2 .. v_k e_k v_{k+1}$ and an interlaced flow f' in H such that $e_k = e$ and $f = f' + f_w$.*
6. *There exists a multiset M of paths of H such that $f = \sum_{w \in M} f_w$.*
7. *Let M be a multiset of paths of H and $f = \sum_{w \in M} f_w$. Then for each $v \in V$*
   (a) $\sum_{e^s = v} \boldsymbol{bb}(e) + \boldsymbol{bf}(e) = |\{w \in M | v \text{ is the first vertex of } w\}|$
   (b) $\sum_{e^t = v} \boldsymbol{bb}(e) + \boldsymbol{pb}(e) = |\{w \in M | v \text{ is the last vertex of } w\}|$

Proof. Items 1-3 follow from the definitions of interlaced flow (5.1) and of unit flow (5.2). Items 4.a and 4.b follow from the definition of unit flow. For the proof of one direction of item 5.a, suppose that there is a path $w = v_1 e_1 v_2 ... v_k e_k v_{k+1}$ where $e_1 = e$, $f = f' + f_w$ and f' is interlaced. Then by item 4.*a* $\boldsymbol{bb}(e) + \boldsymbol{bf}(e)$ must be greater than 1, since $\boldsymbol{bb}'(e) + \boldsymbol{bf}'(e) \geq 0$. For the other direction, suppose $\boldsymbol{bb}(e) + \boldsymbol{bf}(e) \geq 1$. If $\boldsymbol{bb}(e) \geq 1$ then the path has length 1 and

consists of e itself. Now let $\boldsymbol{bb}(e) = 0$ and $\boldsymbol{bf}(e) \geq 1$. Then by the equation of Definition 5.1, there exists an edge e_2 in H with $e^t = e_2^s$ for which $\boldsymbol{pb}(e_2) + \boldsymbol{pf}(e_2) \geq 1$. If $\boldsymbol{pb}(e_2) \geq 1$ take $w = (e^s)e(e^t)e_2(e_2^t)$. Otherwise also by the equation of Definition 5.1 there exists an edge e_3 with $e_2^t = e_3^s$ and $\boldsymbol{pb}(e_3) + \boldsymbol{pf}(e_3) \geq 1$ and so on. For some k we must have $\boldsymbol{pb}(e_k) \geq 1$ since H is finite. This e_k will be the last edge of the path. The proof of item 5.b is analogous. For the first direction we use 4.b instead of 4.a. For the other direction we construct the path w in the reverse order, starting from the last edge. Item 6 follows from items 5.a and 5.b: decompose f successively until the zero interlaced flow is reached. Indeed, item 5 assures that while the flow is non-zero, such decomposition is possible. Item 7.a follows from items 5.a and 4.a and item 7.b from 5.b and 4.b.

5.1 Characterization of Causal Orders and Executions in Terms of *p*-interlaced Flows.

Next, in Definition 5.4 we define a special type of interlaced flow, which we will use in Theorem 5.5 to characterize Hasse diagrams of p/t-net executions and causal orders.

Definition 5.4 (p-interlaced flow). *Let $N = (P, T)$ be a p/t-net, $H = (V, E, l)$ a Hasse diagram with $l : V \to T$ and $p \in P$ a place of N. Then a p-interlaced flow is an interlaced flow $f : E \to \mathbb{N}^4$ which satisfies the two following additional equations around each vertex:*

$$(IN) \qquad In(v) = \sum_{e^t = v} \boldsymbol{bb}(e) + \boldsymbol{pb}(e) = \hat{p}(l(v))$$

$$(OUT) \qquad Out(v) = \sum_{e^s = v} \boldsymbol{bb}(e) + \boldsymbol{bf}(e) = \check{p}(l(v))$$

As we will show in Theorem 5.5 below, executions and causal orders of p/t-nets are intimately connected to p-interlaced flows. We will prove that whenever a Hasse diagram H induces an execution of a given p/t-net N, a set of p-interlaced flows may be associated to it, one flow for each place of N. It turns out that the converse is also true: if we are able to associate such a set of flows to a Hasse diagram H then its induced partial order is an execution of N. A similar result holds with respect to Hasse diagrams of causal orders of N. The only difference is that if an edge belongs to the Hasse diagram of a causal order of N, then it must represent a token that was transmitted from the event that labels its source vertex to the event that labels its target vertex, by using at least one place p as a channel. Thus in the flow that corresponds to p, the component which is responsible for the direct transmission of tokens must be strictly greater than zero. We are ready to state our *interlaced flow theorem*:

Theorem 5.5 (Interlaced Flow Theorem). *Let $N = (P, T)$ be a (not necessarily bounded) p/t-net and $H = (V, E, l)$ be a Hasse diagram. Then*

(*i*) *The partial order induced by H is an execution of N iff there exists a p-interlaced flow $f_p : E \to \mathbb{N}^4$ in H for each place p.*

(*ii*) *The partial order induced by H is a causal order of N iff there exists a set $\{f_p\}_{p \in P}$ of p-interlaced flows such that for every edge e of H, the component $\boldsymbol{bb}_p(e)$ of $f_p(e)$, which denotes the direct transmission of tokens, is strictly greater than zero for at least one $p \in P$.*

Proof. The proof is constructive. Suppose we are given a process π of N and the Hasse diagram H of one of its executions, i.e., a sequentialization of po_π. Then we will construct a set of p-interlaced flows in H, one for each place of N. This will prove one direction of item (*i*). In order to prove the same direction of item (*ii*), we will notice that if H is indeed the Hasse diagram

of po_π itself, then the constructed flows satisfy the additional restriction concerning the direct transmission of tokens. Conversely, provided we are given a Hasse diagram H and a set of p-interlaced flows on it, we will construct a process π of N and show that H is the Hasse diagram of some sequentialization of po_π, proving in this way the other direction of item (i). In order to prove the same direction of item (ii) we will notice that if the flows satisfy the additional requirements concerning the direct transmission of tokens, then H is indeed the Hasse diagram of po_π. The details follow.

From Processes to Flows

(i) Let $\pi = (B \dot\cup V, F, \rho)$ be a process of N and suppose that $H^* = (V, <, l)$ is an execution of N that is a sequentialization of the causal order po_π derived from π. For each place $p \in N$ we extend the functions $\hat p, \check p : T \to \mathbb{N}$ to $\check p : T \cup \{\iota, \epsilon\} \to \mathbb{N}$ and $\hat p : T \cup \{\iota\} \to \mathbb{N}$ by making $\check p(\iota) = p_0$, $\check p(\epsilon) = 0$ and $\hat p(\iota) = 0$. $\hat p(\epsilon)$ represents the number of tokens in p after the occurrence of all events of π. For each $b \in B$ with $(v, b), (b, v') \in F$ for some $v, v' \in V$ we choose an arbitrary path w_b in H (not in H^*) from v to v'. Since H is the Hasse diagram of a sequentialization of the causal order po_π of π, such a path always exists. We claim that for each $p \in P$, $f_p = \sum_{\rho(b)=p} f_{w_b}$ is a p-interlaced flow of N. By Lemma 5.3.2, each f_{w_b} is interlaced. Thus by Lemma 5.3.3, f_p is interlaced as well. By the definition of process (3.1), for each $v \in V$ we have $|\{(v, b) \in F : \rho(b) = p\}| = \check p(\rho(v))$ and thus exactly $\check p(l(v))$ chosen paths whose first vertex is v. It implies, by Lemma 5.3.4 that $\sum_{e^s=v} \boldsymbol{bb}_p(e) + \boldsymbol{bf}_p(e) = \check p(l(v))$. Thus condition (OUT) of Definition 5.4 is satisfied. Analogously $|\{(b, v) \in F : \rho(b) = p\}| = \hat p(\rho(v))$ and thus exactly $\hat p(l(v))$ chosen paths whose last vertex is v, what implies $\sum_{e^t=v} \boldsymbol{pb}_p(e) + \boldsymbol{bb}_p(e) = \hat p(l(v))$. Thus condition (IN) of Definition 5.4 is satisfied as well.

(ii) Suppose that $H = (V, E, l)$ is the Hasse diagram of the causal order po_π derived from π, and let $e = (v, v') \in E$ be one of its edges. Then for some $p \in P$ there exists a condition b in π whose label is p for which $(v, b), (b, v') \in F$. Thus $f_{w_b}(e) = (1, 0, 0, 0)$ which implies that $\boldsymbol{bb}_p(e) \geq 1$. In other words, the component of f_p relative to the direct transmission of tokens is strictly greater than zero.

From Flows to Processes

(i) Suppose that there is a p-interlaced flow $f_p : E \to \mathbb{N}^4$ in H for each $p \in P$. We construct a process $\pi = (B \dot\cup V, F, \rho)$ of N for which $H^* = (V, <, l)$ is one of its executions. First we set $\rho(v) = l(v)$ for each $v \in V$. By Lemma 5.3.6, for each f_p, there exists a multiset M_p of paths of H for which $f_p = \sum_{w \in M_p} f_w$. For each path $w = v_1 e_1 v_2 ... v_k e_k v_{k+1}$ in M_p we create a condition b_w in B labeled by p, i.e. $\rho(b_w) = p$, and put (v_1, b_w) and (b_w, v_{k+1}) into F. We claim that for each $v \in V$, $|\{(b, v) \in F : \rho(b) = p\}| = \hat p(\rho(v))$ and $|\{(v, b) \in F : \rho(b) = p\}| = \check p(\rho(v))$. Since each f_p is a p-interlaced flow, we have $\sum_{e^t=v} \boldsymbol{pb}_p(e) + \boldsymbol{bb}_p(e) = \hat p(l(v))$ and $\sum_{e^s=v} \boldsymbol{bb}_p(e) + \boldsymbol{bf}_p(e) = \check p(l(v))$ for each $v \in V$. By Lemma 5.3.7, for each $p \in P$, there exist exactly $\check p(l(v))$ paths in M_p whose first vertex is v and exactly $\hat p(l(v))$ paths in M_p whose last vertex is v. Furthermore for $v = v_\iota$, there are $\check p(\iota) = p_0$ minimal conditions, which correspond to paths whose first vertex is v_ι.

It remains to check that H^* is indeed a sequentialization of the causal order $po_\pi = (V, <_\pi, l)$ derived from π. Notice that as an alternative to definition (Definition 3.2), we can define the causal order of a process po_π as the transitive closure of the DAG $G_\pi = (V, E', l)$ where $E' = \{(v, v') | (v, b) \in F \text{ and } (b, v') \in F\}$. Thus, in order to show that $<_\pi \subseteq <$, it is enough to show that E' is included in $<$. Let $(v, v') \in E'$ and $(v, b), (b, v') \in F$ for some b in B. Then by our construction of π this condition $b = b_w$ corresponds to a path w in H whose first vertex is v and last is v', what implies that $v < v'$ in H^* and proves the claim.

(*ii*) Suppose that for each $e = (v, v') \in E$ it holds that $\boldsymbol{bb}_p(e) \geq 1$ for at least one place $p \in P$. Then by the construction of process described above, we have a condition b_w such that $(v, b_w), (b_w, v') \in F$. This implies that the edge (v, v') is also in the causal order derived from π. Thus $E \subseteq <_\pi$. Since $<$ is the transitive closure of E, we have $< \subseteq <_\pi$. Since the inclusion in the other direction was already proved, we have $< = <_\pi$.

6 Seasoners: Slicing DAG properties.

Any DAG can be cast as the composition of a sequence of unit slices. Conversely, we may use slice graphs in order to compose unit slices and define infinite families of *DAGs*. Using the notion of *seasoner* which we develop in this section, we will be able to "slice" some graph properties, and transpose them to unit slices. This technique will allow us to define in the next section slice graphs whose graph language consists exclusively of *DAGs* satisfying a given property, and to state our main results in terms of them. In particular we will provide seasoners that characterize the structure of Hasse diagrams (Lemma 6.8), as well as seasoners that characterize Hasse diagrams of causal orders and executions of a given p/t-net (Lemma 6.2).

A *seasoning* of a slice $\mathbf{S} = (V, E, l)$ is a relation $R : E^\alpha \times X$ that associates values of an arbitrary fixed set X to α-tuples of edges of $\mathbf{S}$. A *seasoned slice* is a pair $(\mathbf{S}, R)$ where $\mathbf{S}$ is a slice and R a seasoning of $\mathbf{S}$. A seasoned slice $(\mathbf{S}, R)$ can be composed with a seasoned slice $(\mathbf{S}', R')$ if $\mathbf{S}$ can be composed with $\mathbf{S}'$ and furthermore the values associated by R to each α-tuple of out-edges of $\mathbf{S}$ agree with the values associated by R' to the corresponding α-tuple of in-edges of $\mathbf{S}'$ (figs. 3-*iii* and 4). A seasoning of a sequence $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ of slices is a sequence of relations $R_1R_2...R_n$ where R_i is a seasoning of $\mathbf{S}_i$ for $1 \leq i \leq n$ and such that $(\mathbf{S}_i, R_i)$ can be composed with $(\mathbf{S}_{i+1}, R_{i+1})$ for $1 \leq i \leq n-1$. In this paper we restrict our seasonings to partial functions (viewed as relations).

Definition 6.1 (Seasoner). *A* seasoner *is a decidable second order predicate $Q(\mathbf{S}, R)$ in which the first variable $\mathbf{S}$ ranges over unit slices, the second R over seasonings of $\mathbf{S}$, and for each $\mathbf{S}$ the set $\{R|Q(\mathbf{S}, R)\}$ is finite and computable. A sequence of unit slices $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ is Q-seasonable, if $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ has a seasoning $R_1R_2...R_n$ satisfying $Q(\mathbf{S}_i, R_i)$ for each i. A DAG G is Q-seasonable if each of its unit decompositions $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ is Q-seasonable. A seasoner Q is coherent if for every DAG G, the Q-seasonability of a unit decomposition $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ of G implies the Q-seasonability of G.*

In the next lemma we give our first application of seasoners, namely a local characterization of Hasse diagrams whose induced partial orders are executions or causal orders of a given bounded p/t-net.

Lemma 6.2 (Slicing Hasse Diagrams of Executions and Causal Orders). *Let H be a Hasse diagram with unique minimal and maximal vertices, N a bounded p/t-net with bound k and p a place of N. Then there exist coherent seasoners Q^k_p, Q^N_{ex}, Q^N_{cau} such that*

(i) H has a p-interlaced flow iff H is Q^k_p-seasonable.
(ii) The PO induced by H is an execution of N iff H is Q^N_{ex}-seasonable.
(iii) The PO induced by H is a causal order of N iff H is Q^N_{cau}-seasonable.

Below we explicitly define the seasoners Q^k_p, Q^{ex}_p and Q^{cau}_p. The proof that these seasoners satisfy Lemma 6.2 will be provided in the end of this subsection, after we have stated Proposition 6.4 and Lemma 6.5.

Definition 6.3. *Let $N = (P, T)$ be a bounded p/t-net with bound k, p a place of N and $\mathbf{S} = (V, E, l)$ a standard unit slice with $V = I \dot\cup \{v\} \dot\cup O$ and $l(v) = t$ for some $t \in T$. Then*

1. *(p-seasoner) Define* $Q_p^k(\mathbf{S}, R)$ *to be true if* R *is a function* $R : E \to \mathbb{N}^4$ *where for each* $e \in E$, $R(e) = (\boldsymbol{bb}(e), \boldsymbol{bf}(e), \boldsymbol{pb}(e), \boldsymbol{pf}(e))$ *and*
   (a) $\sum_{e^t=v} \boldsymbol{pb}(e) + \boldsymbol{bb}(e) = \hat{p}(l(v))$,
   (b) $\sum_{e^s=v} \boldsymbol{bb}(e) + \boldsymbol{bf}(e) = \check{p}(l(v))$,
   (c) $\sum_{e^t=v} \boldsymbol{bf}(e) + \boldsymbol{pf}(e) = \sum_{e^s=v} \boldsymbol{pb}(e) + \boldsymbol{pf}(e)$,
   (d) $\sum_{e^s\in I} \boldsymbol{bb}(e) + \boldsymbol{bf}(e) + \boldsymbol{pb}(e) + \boldsymbol{pf}(e) \leq k$,
   (e) $\sum_{e^t\in O} \boldsymbol{bb}(e) + \boldsymbol{bf}(e) + \boldsymbol{pb}(e) + \boldsymbol{pf}(e) \leq k$.
2. *(Execution seasoner) Define* $Q_{ex}^N(\mathbf{S}, R)$ *to be true if* R *is a function* $R : P \to (E \to \mathbb{N}^4)$ *where for each* $p \in P$, $Q_p^k(\mathbf{S}, R(p))$ *holds. For each* $e \in E$ *we let*

$$R(p)(e) = (\boldsymbol{bb}_p(e), \boldsymbol{bf}_p(e), \boldsymbol{pb}_p(e), \boldsymbol{pf}_p(e)).$$

3. *(Causality seasoner) Define* $Q_{cau}^N(\mathbf{S}, R)$ *to be true iff* $Q_{ex}^N(\mathbf{S}, R)$ *holds and if for each* $e \in E$ *there exists a place* $p \in P$ *for which* $\boldsymbol{bb}_p(e) \geq 1$.

A very intuitive account of Q_p^k is the following: Let $H = (V, E, l)$ be a Hasse diagram and $f_p : E \to \mathbb{N}^4$ be a p-flow on H with respect to a k-bounded p/t-net N. Then we may cut H into several unit slices and keep the flow value $f_p(e)$ into each sliced part of an edge $e \in E$, as exemplified in Figure 3.*iii*. Now let $\mathbf{S} = (v, E', l')$ be one of the slices of H and $R : E' \to \mathbb{N}^4$ be a function on the edges of $\mathbf{S}$ such that $R(e') = f_p(e)$ if and only if the edge $e' \in E'$ is one of the sliced parts of e. In other words R may be regarded as a "sliced" part of the function. Since f_p is a p-interlaced flow, R locally behaves as an interlaced flow. Thus R satisfies condition (c), which is precisely the interlaced equation of Definition 5.1 as well as conditions (a) and (b), which are respectively the (IN) and (OUT) equations of Definition 5.4. Since by assumption, the place p belongs to a net N with bound k, R must satisfy conditions (e) and (f) as well. This last fact will be clearer after the analysis of Proposition 6.4 below and of Lemma 6.5.

Proposition 6.4. *Let* $N = (P, T)$ *be a bounded* p/t*-net with bound* k, p *a place of* N, $\mathbf{S} = (V, E, l)$ *a unit slice with* $V = I \dot{\cup} \{v\} \dot{\cup} O$ *and* R *a* Q_p^k*-seasoning of* $\mathbf{S}$. *Then*

$$\sum_{e^t\in O} (\boldsymbol{bb} + \boldsymbol{bf} + \boldsymbol{pb} + \boldsymbol{pf})(e) = \check{p}(l(v)) - \hat{p}(l(v)) + \sum_{e^s\in I} (\boldsymbol{bb} + \boldsymbol{bf} + \boldsymbol{pb} + \boldsymbol{pf})(e).$$

Proposition 6.4 follows from the interlaced equation of Definition 5.1 and from the (IN) and (OUT) equations of Definition 5.4. The next lemma, whose proof is a consequence of Proposition 6.4, says that if $H = \mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n$ is the Hasse diagram of an execution of a p/t-net N with bound k, and if R_i is a Q_p^k seasoning of $\mathbf{S}_i$, then the sum of all entries of the flow values associated to the edges touching the out-frontier of $\mathbf{S}_i$ equals the number of tokens in p after the occurrence of all transitions that label the center vertices of $\mathbf{S}_j$ for $1 \leq j \leq i$.

Lemma 6.5. *Let* $N = (P, T)$ *be a bounded* p/t*-net with bound* k, p *a place of* N , $H = (V, E, l)$ *be a Hasse diagram of an execution of* N, $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ *a unit decomposition of* H *where* $\mathbf{S}_i = (V_i, E_i, l_i)$ *with* $V_i = I_i \cup \{v_i\} \cup O_i$ *and* $R_1R_2...R_n$ *a* Q_p^k*-seasoning of* $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$. *Then for* $1 \leq i \leq n-1$ *the following equality holds:*

$$\sum_{e^t\in O_i} (\boldsymbol{bb}_i + \boldsymbol{bf}_i + \boldsymbol{pb}_i + \boldsymbol{pf}_i)(e) = \sum_{1\leq j\leq i} (\check{p}(l_j(v_j)) - \hat{p}(l_j(v_j)))$$

Proof. The proof is by induction on i. Since the partial order induced by H is an execution of N, H has a unique minimal vertex which is labeled by ι. Thus the in-frontier of $\mathbf{S}_1$ is empty and its unique center vertex is v_ι which is labeled by ι. By equation (c) and (b) of Definition 6.3.1 (p-seasoner), $\sum_{e^s=v_\iota} (\boldsymbol{bb}_1 + \boldsymbol{bf}_1 + \boldsymbol{pb}_1 + \boldsymbol{pf}_1)(e) = \sum_{e^s=v_\iota} (\boldsymbol{bb} + \boldsymbol{bf})(e) = \check{p}(\iota) = m_0(p)$. Thus

the lemma is valid for the base case. Now suppose it is valid for i with $1 \leq i < n-1$. We prove it is valid for $i+1$ as well. By the rule of composition of seasoned slices, the sum over the edges touching the in-frontier of $\mathbf{S}_{i+1}$ must be equal to the sum over the edges touching out-frontier of $\mathbf{S}_i$. Thus we have

$$\sum_{e^s \in I_{i+1}} (\boldsymbol{bb}_{i+1} + \boldsymbol{bf}_{i+1} + \boldsymbol{pb}_{i+1} + \boldsymbol{pf}_{i+1})(e) = \sum_{1 \leq j \leq i} (\check{p}(l_j(v_j)) - \hat{p}(l_j(v_j))).$$

This equation together with Proposition 6.4 imply that

$$\sum_{e^t \in O_{i+1}} (\boldsymbol{bb}_{i+1} + \boldsymbol{bf}_{i+1} + \boldsymbol{pb}_{i+1} + \boldsymbol{pf}_{i+1})(e) = \sum_{1 \leq j \leq i+1} (\check{p}(l_j(v_j)) - \hat{p}(l_j(v_j)))$$

what proves the lemma.

The seasoner Q^N_{ex} of Definition 6.3.2 is built on base of Q^k_p. Instead of associating a single p-interlaced flow value for each edge of a unit slice, it associates tuples of p-interlaced flows to these edges, one flow for each place p of N. In this way we transpose the characterization of Hasse diagrams of p/t-net executions to unit slices. The intuition is that whenever the composition of a sequence of Q^N_{ex} seasoned slices yields a Hasse diagram with unique minimal and maximal vertices, it yields as well a set of p-interlaced flows, one for each place of N. Thus by Theorem 5.5.(i) it follows that the partial order induced by H is an execution of N. If in addition the component $\boldsymbol{bb}_p(e) \geq 1$ for at least one of the p-interlaced flow values associated to each edge e of the slice, as required by the Q^N_{cau} seasoner of Definition 6.3.3, then Theorem 5.5.(ii) implies that the partial order induced by H is indeed a causal order of N. Below we provide a formal proof of Lemma 6.2, which summarizes the discussion of this subsection.

Proof of Lemma 6.2 Let $N = (P, T)$ be a p/t-net with bound k, $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ be a unit decomposition of a Hasse diagram H and $R_1R_2...R_n$ be a Q^k_p-seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ with respect to a place $p \in P$. Then $(\mathbf{S}, R_1) \circ (\mathbf{S}_2, R_2) \circ ... \circ (\mathbf{S}_n, R_n)$ is equal to H with a p-interlaced flow associated to its edges (Fig. 3.iii). Conversely, if it is possible to associate a p-interlaced flow f_p to a Hasse diagram H, then to any of its unit decompositions $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$, we can associate a seasoning $R_1R_2...R_n$ where $R_i(e^i) = f_p(e)$ if and only if e^i is the sliced part of e which lies on $\mathbf{S}_i$ (Fig. 3.iii). Since f_p is interlaced and satisfies conditions (IN) and (OUT) of Definition 5.4, equations (a),(b) and (c) of Definition 6.3 are satisfied. Since N is bounded by k, Lemma 6.5 assures that equations (d) and (b) of Definition 6.3.1 are satisfied as well. This last claim follows from the fact that $\sum_{1 \leq j \leq i}(\check{p}(l_j(v_j)) - \hat{p}(l_j(v_j)))$ corresponds to the number of tokens at place p after the firing of transitions $l_1(v_1)l_2(v_2)...l_k(v_i)$.

If $R_1R_2...R_n$ is a Q^N_{ex}-seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$, then $(\mathbf{S}, R_1) \circ (\mathbf{S}_2, R_2) \circ ... \circ (\mathbf{S}_n, R_n)$ is equal to H with a tuple of p-interlaced flows associated to its edges, one flow for each place p of N, and thus by Theorem 5.5.(i), the partial order induced by H is an execution of N. If in addition $R_1R_2...R_n$ is a Q^N_{cau} seasoning of N, then by Theorem 5.5.(ii), H is the Hasse diagram of a causal order of N. Conversely, if it is possible to associate a tuple $(f_{p_1}, f_{p_2}, ..., f_{p_{|P|}})$ of interlaced flows to H, then by letting $R_i(e^i) = (f_{p_1}(e), f_{p_2}(e), ..., f_{p_{|P|}}(e))$ iff e^i is the sliced part of e which lies in $\mathbf{S}_i$, we have that $R_1R_2...R_n$ is an execution seasoning of H. If in addition, for at least one $p_r \in P$ the component $\boldsymbol{bb}_{p_r}(e)$ of $f_{p_r}(e)$ is strictly greater than zero for each edge e of H, then $R_1R_2...R_n$ is a causal seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$. Finally, since the discussion in this proof is completely independent of the way in which H is sliced, all the three seasoners Q^k_p, Q^N_{ex} and Q^N_{cau} are coherent. □

6.1 Slicing Hasse Diagrams

Our second example of seasoner, the *Hasse seasoner* introduced below in Definition 6.6 will allow us to characterize the structure of Hasse diagrams at the slice level. In other words, by using Hasse seasoners we will "season" the edges of unit slices with some additional information in such a way that the coherent composition of these new seasoned slices is guaranteed to yield Hasse diagrams. This result will be formally stated in Lemma 6.8 and will be of particular importance for our expressibility theorem (Theorem 7.8), which shows that the behavior of bounded p/t-net can be adequately captured by Hasse diagram generators.

Definition 6.6 (Hasse Seasoning and Hasse Seasoner). *Let $\mathbf{S} = (V, E, l)$ be a unit standard slice whose unique center vertex is v. Then a partial function $\mathcal{H} : E^2 \to \{0,1\}^2$ is a* Hasse seasoning *of $\mathbf{S}$ if the following conditions can be verified for every $e_1, e_2 \in E$:*

1. *$\mathcal{H}(e_1e_2)$ is not defined if and only if $(e_1{=}e_2)$ or $(e_1^t{=}e_2^s)$ or $(e_1^s{=}e_2^t)$,*
2. *if $\mathcal{H}(e_1e_2) = xy$ then $\mathcal{H}(e_2e_1) = yx$ (for $x, y \in \{0,1\}$),*
3. *if $e_1^t{=}e_2^t$ then $\mathcal{H}(e_1e_2) = 11$,*
4. *if $e_1^s{=}e_2^s$ then $\mathcal{H}(e_1e_2) = 00$,*
5. *if $e_1^s \in I$ and $e_1^t \in O$ and $e_2^s = v$ then $\mathcal{H}(e_1e_2) \in \{01, 11\}$ and*
$$\mathcal{H}(e_1e_2) = 01 \text{ iff } (\exists e, e^t{=}v)(\mathcal{H}(e_1e) \in \{00, 01\}).$$

We define the Hasse seasoner $Q_{\mathcal{H}}(\mathbf{S}, R)$ to be true if $\mathbf{S}$ is a unit slice and R is a Hasse seasoning of $\mathbf{S}$. See figure 4.

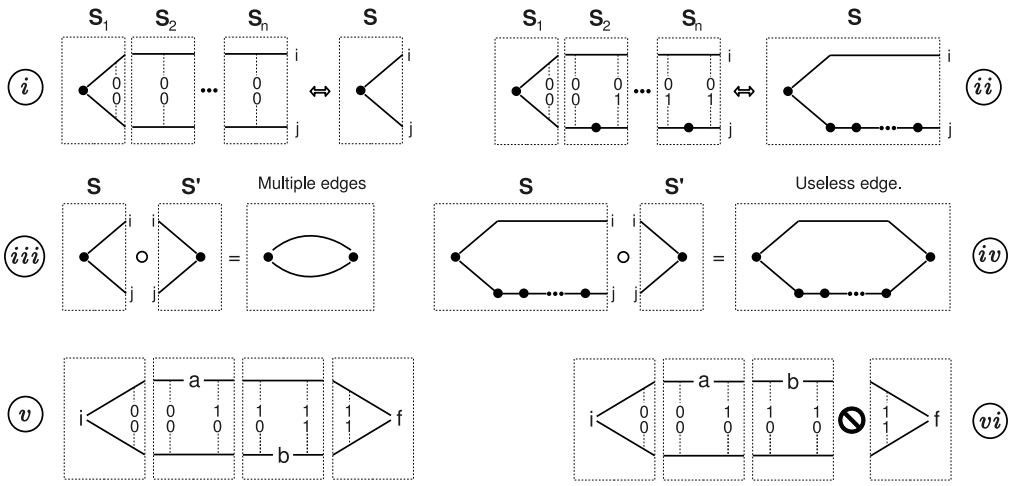

Fig. 4. *i*) i, j diverge in $\mathbf{S}$. *ii*) i is shorter than j in $\mathbf{S}$. *iii*) and *iv*) i, j converge in $\mathbf{S}'$. *v*) Any unit decomposition of a Hasse diagram has a Hasse seasoning. *vi*) If a *DAG* is not a Hasse diagram, none of its unit decompositions can be coherently Hasse-seasoned.

In this section in particular, it will be convenient to write simply j for the unique edge touching a frontier vertex v of a slice $\mathbf{S} = (V, E, l)$ in which $l(v) = j$. The context will always clarify whether v is an *in* or an *out* frontier vertex. In this setting we write j^s to indicate the source vertex of j and j^t to indicate its target vertex. We say that two edges i, j of $\mathbf{S}$ *converge* if they touch the in-frontier of $\mathbf{S}$ and $i^t = j^t$ (Fig. 4.*iv*). We say i, j *diverge* if they touch the out-frontier of $\mathbf{S}$ and $i^s = t^s$ (Fig. 4.*i*) . Finally we say that i is shorter than j if they touch the out-frontier of $\mathbf{S}$ and if there is a path with at least one edge from i^s to j^s (Fig. 4.*ii*). Below in Proposition 6.7, we state some easily verifiable facts about Hasse seasonings which will be used in the proof of Lemma 6.8.

Proposition 6.7. *1. Let $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ be a unit decomposition of a slice $\mathbf{S}$ with a unique minimal vertex, $\mathcal{H}_1\mathcal{H}_2...\mathcal{H}_n$ a Hasse seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ and $i, j \in O^n$ be two out-frontier vertices of $\mathbf{S}_n$. Then*
- *$\mathcal{H}_n(i,j) = 00 \Leftrightarrow i$ and j diverge in $\mathbf{S}$. See Fig. 4-i.*
- *$\mathcal{H}_n(i,j) = 01 \Leftrightarrow i$ is shorter than j in $\mathbf{S}$. See Fig. 4-ii.*

2. Let $\mathbf{S}, \mathbf{S}'$ be two (not necessarily unit) slices such that $\mathbf{S}$ can be composed with $\mathbf{S}'$. And let i, j be two convergent in-frontier vertices in $\mathbf{S}'$. If i, j are divergent in $\mathbf{S}$ or if i is shorter than j in $\mathbf{S}$ then $\mathbf{S} \circ \mathbf{S}'$ is not transitive reduced. See Figs. 4-iii and 4-iv.

3. Let $\mathbf{S}$ and $\mathbf{S}'$ be two unit slices such that $\mathbf{S}$ can be composed with $\mathbf{S}'$ and $\mathcal{H}$ be a Hasse seasoning of $\mathbf{S}$. If there is no Hasse seasoning $\mathcal{H}'$ of $\mathbf{S}'$ such that $(\mathbf{S}, \mathcal{H})$ can be composed with $(\mathbf{S}', \mathcal{H}')$, then there exist i, j such that $\mathcal{H}(i,j) \in \{00, 01\}$ and i, j converge in $\mathbf{S}'$. See Fig. 4-vi.

Lemma 6.8 (Seasoners and Hasse Diagrams). *Let H be a Hasse diagram, G a DAG and $Q_{\mathcal{H}}$ be the Hasse seasoner defined in 6.6. Then G is a Hasse diagram with a unique minimal vertex iff G is $Q_{\mathcal{H}}$-seasonable.*

Proof. Let $\mathbf{S} = \mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n$ be a slice with a unique minimal vertex. Suppose that $\mathbf{S}$ is not a Hasse diagram and that it is $Q_{\mathcal{H}}$-seasonable. Let $\mathcal{H}_1\mathcal{H}_2...\mathcal{H}_n$ be a $Q_{\mathcal{H}}$-seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$. Since $\mathbf{S}$ is not a Hasse diagram, there exist two vertices v and v' which are connected either by multiple edges, or by an edge and a path of length greater than one. Let v' be in $\mathbf{S}_{k+1}$ for some $1 \leq k \leq n-1$. Then there are i, j such that i, j converge in $\mathbf{S}_{k+1}$. Furthermore either i, j diverge or i is shorter than j in $\mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_k$. By proposition 6.7-1, we have that $\mathcal{H}_k(i,j) \in \{00, 01\}$. By proposition 6.7-3, there is no $\mathcal{H}_{k+1}$ such that $(\mathbf{S}_k, \mathcal{H}_k)$ can be composed with $(\mathbf{S}_{k+1}, \mathcal{H}_{k+1})$. This contradicts the assumption that $\mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n$ is $Q_{\mathcal{H}}$-seasonable.

In order to prove the converse, let $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_k$ be the greatest prefix of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ which is $Q_{\mathcal{H}}$-seasonable and let $\mathcal{H}_1\mathcal{H}_2...\mathcal{H}_k$ be one of its Hasse seasonings. Suppose that $k < n$. By Proposition 6.7-3, there exist integers i, j such that $\mathcal{H}_k(i,j) \in \{00, 01\}$ and i, j converge in $\mathbf{S}_{k+1}$. By proposition 6.7-1, either i, j diverge or i is shorter than j in $\mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_k$. This implies, by Proposition 6.7-2 that $\mathbf{S}_1 \circ ... \circ \mathbf{S}_k \circ \mathbf{S}_{k+1}$ is not transitive reduced, and so neither is $\mathbf{S}_1 \circ ... \circ \mathbf{S}_n$.

7 Main Results

In order to provide a unified framework for the statement of our main results, we introduce the notion of *filter*. A filter is a function $filter(\mathcal{SG}, Q)$ that takes as input a slice graph $\mathcal{SG}$ and a coherent seasoner Q and returns a slice graph $\mathcal{SG}'$ whose graph language $\mathcal{L}_G(\mathcal{SG}')$ is formed exclusively by the *DAGs* in $\mathcal{L}_G(\mathcal{SG})$ that are Q-seasonable. In other words, it filters out from $\mathcal{L}_G(\mathcal{SG})$ every *DAG* which is not Q-seasonable.

In particular, if $Q_{\mathcal{H}}$ is the Hasse seasoner introduced in Definition 6.6, then $filter(\mathcal{SG}, Q_{\mathcal{H}})$ is a Hasse diagram generator that generates precisely the Hasse diagrams in $\mathcal{L}_G(\mathcal{SG})$. Analogously, given a p/t-net N, if Q^N_{ex} (Q^N_{cau}) is the execution (causal) seasoner of Definition 6.3 and $\mathcal{HG}$ is a Hasse diagram generator, then the graph language of $filter(\mathcal{HG}, Q^N_{ex})$ ($filter(\mathcal{HG}, Q^N_{cau})$) contains exactly the Hasse diagrams generated by $\mathcal{HG}$ whose induced partial orders are executions (causal orders) of N.

Definition 7.1 (Filter). *Let $\mathcal{SG} = (\mathcal{V}, \mathcal{E}, \mathcal{S})$ be a slice graph where $\mathcal{S} : V \to \Sigma_{\mathbb{S}}$ and let Q be a coherent seasoner. Then the Q-filter of $\mathcal{SG}$ is the slice graph $filter(\mathcal{SG}, Q) = (\mathcal{V}^f, \mathcal{E}^f, \mathcal{S}^f)$ where*

$$\mathcal{V}^f = \bigcup_{v \in \mathcal{V}} \{v_R | Q(\mathcal{S}(v), R)\} \qquad \mathcal{S}^f : \mathcal{V}^f \to \Sigma_{\mathbb{S}} \qquad \mathcal{S}^f(v_R) = \mathcal{S}(v)$$

$$\mathcal{E}^f = \{(v_R, v'_{R'}) | (v, v') \in \mathcal{E} \text{ and } (\mathcal{S}(v), R) \text{ can be composed with } (\mathcal{S}(v'), R')\}$$

Intuitively, $filter(\mathcal{SG}, Q)$ takes a slice graph $\mathcal{SG}$ and expands each of its vertices v into a set of vertices $\{v_R | Q(\mathcal{S}(v), R)\}$. Namely, a vertex v_R for each Q-seasoning R of the slice $\mathcal{S}(v)$ that labels v. Each of these new vertices v_R are then labeled with $\mathcal{S}(v)$ as well, i.e. $\mathcal{S}^f(v) = \mathcal{S}(v)$. However now a vertex v_R is connected to a vertex $v'_{R'}$ if and only if v was connected to v' in the original slice graph and if the seasoned slice $(S(v), R)$ can be composed with $(S'(v), R')$. As we show in the next lemma, the function we just defined indeed behaves as a filter, in the sense that it filters out from $\mathcal{L}_G(\mathcal{SG})$ all the $DAGs$ that are not Q-seasonable, as described in the beginning of this section.

Lemma 7.2 (Filter Lemma). *Let Q be a coherent seasoner and $\mathcal{SG}$ a slice graph. Then*

1. *$filter(\mathcal{SG}, Q)$ is effectively computable.*
2. *$\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}(filter(\mathcal{SG}, Q))$ if and only if $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}(\mathcal{SG})$ and $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ is Q-seasonable.*
3. *$H \in \mathcal{L}_G(filter(\mathcal{SG}, Q))$ iff $H \in \mathcal{L}_G(\mathcal{SG})$ and H is Q-seasonable.*
4. *$\mathcal{L}_G(\mathcal{SG}) = \mathcal{L}_G(filter(\mathcal{SG}, Q))$ if and only if $\mathcal{L}(\mathcal{SG}) = \mathcal{L}(filter(\mathcal{SG}, Q))$*
5. *It is decidable whether $\mathcal{L}_G(filter(\mathcal{SG}, Q)) = \emptyset$ as well as whether $\mathcal{L}_G(\mathcal{SG}) = \mathcal{L}_G(filter(\mathcal{SG}, Q))$.*

Proof. 1. For each $v \in \mathcal{V}$, the set $\{v_R | Q(\mathcal{S}(v), R)\}$ is finite and computable (Definition 6.1).

2. By definition of filter, the label $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ of a walk $v^1v^2...v^n$ in $\mathcal{SG}$ has a Q-seasoning $R_1R_2...R_n$ iff $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ labels the walk $v^1_{R_1}v^2_{R_2}...v^n_{R_n}$ in $filter(\mathcal{SG}, Q)$.
3. If $H \in \mathcal{L}_G(filter(\mathcal{SG}, Q))$ then there exists a walk $v^1_{R_1}v^2_{R_2}...v^n_{R_n}$ in $filter(\mathcal{SG}, Q)$ whose label is $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}(filter(\mathcal{SG}, Q))$ and such that $H = \mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n$. By item 2, $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ belongs to $\mathcal{L}(\mathcal{SG})$ and consequently $H \in \mathcal{L}_G(\mathcal{SG})$. Furthermore, $R_1R_2...R_n$ is a Q-seasoning of $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$, what implies that H is Q-seasonable. Conversely, if $H \in \mathcal{L}_G(\mathcal{SG})$ then there exists a walk $v^1v^2...v^n$ in $\mathcal{SG}$ whose label is $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ and such that $H = \mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_n$. If H is Q-seasonable, then since Q is coherent, so is $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$. By item 2, $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}(filter(\mathcal{SG}, Q))$ and consequently $H \in \mathcal{L}_G(filter(\mathcal{SG}, Q))$.
4. By definition of graph language (Definition 2.4), if $\mathcal{L}(\mathcal{SG})$ is equal to $\mathcal{L}(filter(\mathcal{SG}, Q))$ then $\mathcal{L}_G(\mathcal{SG})$ is equal to $\mathcal{L}_G(filter(\mathcal{SG}, Q))$ as well. By definition of filter, both $\mathcal{L}(filter(\mathcal{SG}, Q)) \subseteq \mathcal{L}(\mathcal{SG})$ and $\mathcal{L}_G(filter(\mathcal{SG}, Q)) \subseteq \mathcal{L}_G(\mathcal{SG})$. Thus we need only to prove that $\mathcal{L}_G(\mathcal{SG}) \subseteq \mathcal{L}_G(filter(\mathcal{SG}, Q))$ implies $\mathcal{L}(\mathcal{SG}) \subseteq \mathcal{L}(filter(\mathcal{SG}, Q))$. Indeed we prove the contrapositive. Let $H \in \mathcal{L}_G(\mathcal{SG})$ be the composition of a sequence of slices $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ in $\mathcal{L}(\mathcal{SG})$. If $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \in \mathcal{L}(\mathcal{SG})$ is not Q-seasonable then by item 2, $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n \notin \mathcal{L}(filter(\mathcal{SG}, Q))$. Since Q is coherent (Definition 6.1), no other unit decomposition $\mathbf{S}'_1\mathbf{S}'_2...\mathbf{S}'_n$ of H can be Q-seasonable. Again by item 2, no other unit decomposition $\mathbf{S}'_1\mathbf{S}'_2...\mathbf{S}'_n$ of H can be in $\mathcal{L}(filter(\mathcal{SG}, Q))$. Thus H is not in $\mathcal{L}_G(filter\mathcal{SG})$.
5. Follows from item 4 and from the fact that $\mathcal{L}(\mathcal{SG})$ and $\mathcal{L}(filter(\mathcal{SG}, Q))$ are regular languages, for which both equality and emptiness are well known to be decidable.

Now we are in a position to introduce our first main result, the verification theorem, which states that for any bounded p/t-net N, it is decidable whether the partial order language generated by a Hasse diagram generator is included in the partial order language of N, or whether their intersection is empty. The result holds for both the causal and execution semantics.

Theorem 7.3 (Verification Theorem). *Let N be a bounded p/t-net, $\mathcal{HG}$ a Hasse diagram generator and $sem \in \{ex, cau\}$. Then*

(i) $\mathcal{L}_{PO}(\mathcal{HG}) \subseteq \mathcal{L}_{sem}(N) \Leftrightarrow \mathcal{L}_{PO}(\mathcal{HG}) = \mathcal{L}_{PO}(filter(\mathcal{HG}, Q^N_{sem}))$
(ii) $\mathcal{L}_{PO}(\mathcal{HG}) \cap \mathcal{L}_{sem}(N) = \emptyset \Leftrightarrow \mathcal{L}_{PO}(filter(\mathcal{HG}, Q^N_{sem})) = \emptyset$

Furthermore the equalities on the right hand side of the $\Leftrightarrow$ symbol are decidable.

Proof. Since $\mathcal{HG}$ is a Hasse diagram generator, $\mathcal{L}_{PO}(\mathcal{HG}) = \mathcal{L}_{PO}(filter(\mathcal{HG}, Q^N_{sem}))$ iff $\mathcal{L}_G(\mathcal{HG}) = \mathcal{L}_G(filter(\mathcal{HG}, Q^N_{sem}))$ and $\mathcal{L}_{PO}(filter(\mathcal{HG}, Q^N_{sem})) = \emptyset$ iff $\mathcal{L}_G(filter(\mathcal{HG}, Q^N_{sem})) = \emptyset$. By lemma 7.2-5 these equalities are decidable. By Lemma 6.2, the partial order induced by a Hasse diagram H in $\mathcal{L}_G(\mathcal{SG})$ is an execution (causal order) of N if and only if H is Q^N_{ex}-seasonable (Q^N_{cau}-seasonable). Thus, by Lemma 7.2-3, $H \in \mathcal{L}_G(filter(\mathcal{HG}, Q^N_{sem}))$ iff $H \in \mathcal{L}_G(\mathcal{HG})$ and the partial order induced by H is an execution ($sem = ex$) or causal order ($sem = cau$) of N.

Below we define the *slicewidth* of slices and $DAGs$. Our result concerning the expressibility of slicegraphs depends on the fact that the Hasse diagrams of partial orders generated by bounded p/t-nets have bounded slicewidth.

Definition 7.4 (Slicewidth). *We define the* slicewidth $sw(\mathbf{S})$ *of a slice* $\mathbf{S} = (V, E, l)$ *as the size of its greatest frontier, i.e. if* $V = I \dot\cup C \dot\cup O$ *then* $sw(\mathbf{S}) = \max\{|I|, |O|\}$. *The slicewidth of a unit decomposition* $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n$ *is the slicewidth of its widest slice:* $sw(\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n) = \max\{sw(\mathbf{S}_i)\}$. *The slicewidth of a DAG* H *is the slicewidth of its thinnest unit decomposition:* $sw(H) = \min\{sw(\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_n) | \mathbf{S}_1 \circ ... \circ \mathbf{S}_n = H\}$.

We compare our definition of slicewidth with the well known notion of width of a partial order. A chain in a partial order po is a totally ordered subset of elements of po. A cut of po is a set of pairwise non-comparable elements of po. The width $w(po)$ of po is defined as the minimal number of chains in which the elements of po can be partitioned, which by [10] is equal to the size of the largest cut in po. We observe that the slicewidth of the Hasse diagram H of a partial order po is less than or equal to its width, i.e. $sw(H) \leq w(po)$. We also notice that in a bounded p/t-net $N = (P, T)$ with bound k at most $k \cdot |P|$ transitions can be fired concurrently in any marking of N. This implies that the size of a cut in any execution or causal order of N is bounded by $k \cdot |P|$. From this discussion we have the following proposition:

Proposition 7.5. *Let* $N = (P, T)$ *be a bounded p/t-net with bound* k. *Then the slicewidth* $sw(H)$ *of any Hasse diagram whose induced partial order is an execution (or causal order) of* H *is bounded by* $k \cdot |P|$.

The slice graphs we define next will have an important role for our expressibility result, Theorem 7.8. Intuitively they generate all $DAGs$ of a given slicewidth n labeled over a set of transitions T. We state their precise behavior in Proposition 7.7.

Definition 7.6 (Bounded Width Slicegraph). *Let* $n \in \mathbb{N}$. *Then we denote* $\mathcal{SG}_n$ *the slice graph* $(\mathcal{V}, \mathcal{E}, \mathcal{S})$ *over* $\Sigma_{\mathbb{S}}$ *where*

$$\Sigma_{\mathbb{S}} = \{\mathbf{S} | \mathbf{S} \text{ is a unit standard slice over } T \cup \{\iota, \varepsilon\} \ , sw(\mathbf{S}) \leq n\}$$

$$\mathcal{V} = \{v_{\mathbf{S}} | \mathbf{S} \in \Sigma_{\mathbb{S}}\} \qquad \mathcal{S} : \mathcal{V} \to \Sigma_{\mathbb{S}} \qquad \mathcal{S}(v_{\mathbf{S}}) = \mathbf{S}$$

$$\mathcal{E} = \{(v_{\mathbf{S}}, v'_{\mathbf{S}'}) \in \mathcal{V} \times \mathcal{V} \mid \mathbf{S} \text{ can be composed with } \mathbf{S}'\}$$

Proposition 7.7. *A DAG* H *labeled over* $T \cup \{\iota, \varepsilon\}$ *belongs to* $\mathcal{L}_G(\mathcal{SG}_n)$ *if and only if its slicewidth* $sw(H)$ *is upper bounded by* n, *and* H *has a unique minimal vertex labeled by* ι *and a unique maximal vertex labeled by* ε.

Proof. Clearly any DAG $H = \mathbf{S}_1 \circ \mathbf{S}_2 \circ ... \circ \mathbf{S}_k$ generated by $\mathcal{SG}_n$ has slicewidth at most n. The fact that $\mathbf{S}_1$ is initial, $\mathbf{S}_k$ final and all the slices $\mathbf{S}_i$ are standard implies that H has unique minimal and maximal vertices. Conversely if $sw(H) \leq n$ and H has unique minimal and

maximal vertices, then there exists a unit decomposition $\mathbf{S}_1\mathbf{S}_2...\mathbf{S}_k$ of H satisfying $sw(\mathbf{S}_i) \leq n$ for $1 \leq i \leq k$, in which all the slices are standard. By definition of $\mathcal{SG}_n$, each $\mathbf{S}_i$ labels a vertex $v_{\mathbf{S}_i}$ of $\mathcal{SG}_n$ and since $\mathbf{S}_i$ can be composed with $\mathbf{S}_{i+1}$, there is an edge between $v_{\mathbf{S}_i}$ and $v_{\mathbf{S}_{i+1}}$. Hence, $H \in \mathcal{L}_G(\mathcal{SG}_n)$. Furthermore, by our labeling convention, the vertex of a initial slice (the unique minimal vertex of a DAG) is always labeled by ι and the center vertex of a final slice (the unique maximal vertex of a DAG) is always labeled by ε.

Our second main result, the expressibility Theorem 7.8, states that for any given bounded Petri net $N = (P, T)$ there exist effectively computable Hasse diagram generators whose partial order languages match the causal behavior and the execution behaviors of N.

Theorem 7.8 (Expressibility Theorem). *Let $N = (P, T)$ be a bounded p/t-net with bound k, $n = k \cdot |P|$, $\mathcal{HG}_n = filter(\mathcal{SG}_n, Q_{\mathcal{H}})$ and $\mathcal{HG}^N_{sem} = filter(\mathcal{HG}_n, Q_{sem})$. Then $\mathcal{L}_{sem}(N) = \mathcal{L}_{PO}(\mathcal{HG}^N_{sem})$.*

Proof. By Lemma 6.8, H is a Hasse diagram iff it is $Q_{\mathcal{H}}$-seasonable. Thus by Proposition 7.7, and by the filter lemma 7.2-3, H belongs to $\mathcal{L}_G(\mathcal{HG}_n)$ iff H is a Hasse diagram whose slicewidth is bounded by n. By Proposition 7.5 the slicewidth of the Hasse diagram of any execution or causal order of N is upper bounded by n, thus every such Hasse diagram belongs to the graph language of $\mathcal{HG}_n$. Now, by items (ii) and (iii) of Lemma 6.2, H is the Hasse diagram of an execution $(sem = ex)$ or causal order $(sem = cau)$ of N if and only if H is Q_{sem} seasonable. Thus applying again the filter Lemma 7.2-3, we have that $H \in \mathcal{L}_G(filter(\mathcal{HG}_n, Q_{sem}))$ if and only if H is Hasse diagram of an execution $(sem = ex)$ or causal order $(sem = cau)$ of N.

As a corollary we have the decidability of the inclusion of the causal and execution languages of any two given bounded nets N_1, N_2: Using the expressibility Theorem 7.8 we compute $\mathcal{HG}^{N_1}_{sem}$. Then employing the verification Theorem 7.3, we test whether $\mathcal{L}_{PO}(\mathcal{HG}^{N_1}_{sem}) \subseteq \mathcal{L}_{sem}(N_2)$. We note that we don't need to compute the Hasse diagram generator corresponding to N_2 and thus the undecidability result stated in Theorem 2.6 does not prevent us from testing the inclusion of the behavior of two p/t-nets. This corollary will be particularly useful in Section 8 where we consider the synthesis of p/t-nets from Hasse diagram generators carrying the causal semantics.

Corollary 7.9 (Comparison of p/t-nets Partial Order Langauges). *Let N_1, N_2 be two bounded Petri nets. Then it is decidable whether $\mathcal{L}_{sem}(N_1) \subseteq \mathcal{L}_{sem}(N_2)$.*

8 Synthesis

In this section we are concerned with the synthesis of k-safe p/t-nets from Hasse diagram generators. The synthesis problem can be formally stated as follows: Given a Hasse diagram generator $\mathcal{HG}$, a semantics $sem \in \{ex, cau\}$ and $k \in \mathbb{N}$, construct a k-safe p/t-net N whose language $\mathcal{L}_{sem}(N)$ minimally includes $\mathcal{L}_{PO}(\mathcal{HG})$. Here minimal inclusion means that given any other k-safe p/t-net N', the chain of inclusions $\mathcal{L}_{PO}(\mathcal{HG}) \subseteq \mathcal{L}_{sem}(N') \subseteq \mathcal{L}_{sem}(N)$ implies that $\mathcal{L}_{sem}(N) = \mathcal{L}_{sem}(N')$. In this work we do not address the question of whether the behavior of the constructed net actually matches the partial order behavior specified by the slice graph. Our approach for the synthesis problem starts with the definition of k-safe regions for slice graphs.

Definition 8.1 (k-safe Region for Hasse Diagram Generators). *Let $\mathcal{HG}$ be a Hasse diagram generator and T a set of transitions. Then a k-safe region of $\mathcal{HG}$ with relation to T is a place $p = (p_0, \hat{p}, \check{p})$ over T such that $\mathcal{L}(\mathcal{HG}) = \mathcal{L}(filter(\mathcal{HG}, Q^k_p))$. We denote $\mathcal{R}_k(\mathcal{HG})$ the set of all k-safe regions of $\mathcal{HG}$.*

Intuitively, a k-safe region of $\mathcal{HG}$ is a k-safe place for which all partial orders generated by $\mathcal{HG}$ have a p-interlaced flow. We note that the set of k-safe places is finite for a given k. It turns out that the union of all these regions gives rise to a k-safe p/t-net with the unique minimal k-safe behavior with relation to $\mathcal{HG}$.

Theorem 8.2 (Synthesis for the Execution Semantics). *Let $\mathcal{HG}$ be a Hasse diagram generator and $k \geq 1$. Then if $\mathcal{R}_k(\mathcal{HG}) \neq \emptyset$, the net $\mathcal{N}^k_{ex}(\mathcal{HG}) = \bigcup_{p \in \mathcal{R}_k(\mathcal{HG})}(\{p\}, T)$ is a k-safe net with minimal execution behavior with relation to $\mathcal{L}_{PO}(\mathcal{HG})$.*

Proof. Let $p \in \mathcal{R}_k(\mathcal{HG})$. Then by definition 8.1, $\mathcal{L}(\mathcal{HG}) = \mathcal{L}(filter(\mathcal{HG}, Q^k_p))$, and by the filter Lemma 7.2-4, $\mathcal{L}_G(\mathcal{HG}) = \mathcal{L}_G(filter(\mathcal{HG}, Q^k_p))$. Thus by lemmas 7.2-2 and 6.2-1, each $H \in \mathcal{L}_G(\mathcal{HG})$ has a p-interlaced flow. This implies, by Theorem 5.5, that each $H \in \mathcal{L}_G(\mathcal{HG})$ is the Hasse diagram of an execution of $\mathcal{N}^k_{ex}(\mathcal{HG})$, i.e, $\mathcal{L}_{PO}(\mathcal{HG}) \subseteq \mathcal{L}_{ex}(\mathcal{N}^k_{ex})$. The minimality of $\mathcal{L}_{ex}(\mathcal{N}^k_{ex})$ follows from two facts: First, if p is not a region, then $\mathcal{L}_{PO}(\mathcal{HG}) \nsubseteq \mathcal{L}_{ex}(\mathcal{N}^k_{ex} \cup (p, T))$, since $\mathcal{L}_G(\mathcal{HG}) \neq \mathcal{L}_G(filter(\mathcal{HG}, Q^k_p))$ and there is an $H \in \mathcal{L}_G(\mathcal{HG})$ which has no p-interlaced flow. Second, by Lemma 4.1, adding a place to a net can at most restrict its execution behavior, and $\mathcal{N}^k_{ex}$ is the union of all possible regions.

We notice that $\mathcal{N}^k_{ex}$ has minimal behavior among all k-safe petri nets. But since $\mathcal{N}^k_{ex}$ is a subnet of $\mathcal{N}^{k+1}_{ex}$ it follows from Lemma 4.1 that $\mathcal{L}_{ex}(\mathcal{N}^{k+1}_{ex}) \subseteq \mathcal{L}_{ex}(\mathcal{N}^k_{ex})$. The synthesis[6] of bounded p/t-nets from Hasse diagram generators, which is still open, is equivalent to determine the minimal k such that the inclusion also holds in the opposite direction.

With relation to the causal semantics, we impose a bound on the number of copies of each place on the net. We say that a p/t-net N is (k, r)-safe if it is k-safe and if it contains at most r copies of each place of N. This restriction is not necessary for the execution semantics because repeated places do not interfere in the execution behavior of p/t-nets, as pointed out in section 4. However they do interfere on the causal behavior of p/t-nets. Another particularity of the causal semantics is the fact that the uniqueness of the minimal behavior is not assured. Nonetheless, by using Corollary 7.9 we are still able to compute the set of minimal nets.

Theorem 8.3. *Let $\mathcal{HG}$ be a Hasse diagram generator and $k, r \geq 1$. Then the set of (k, r)-safe p/t-nets, with minimal causal behavior with relation to $\mathcal{L}_{PO}(\mathcal{HG})$, is computable.*

Proof. Any (k, r)-safe net whose causal behavior includes $\mathcal{L}_{PO}(\mathcal{HG})$ must be a multiset of k-safe regions of $\mathcal{HG}$ in which each region appears at most r times. Let $\mathcal{M}_{k,r}(\mathcal{HG})$ be the set of all such multisets whose unions give rise to legal nets, i.e., for which every transition in T takes tokens of some place and puts on some other. Since $\mathcal{R}_k(\mathcal{HG})$ is finite, so is $\mathcal{M}_{k,r}(\mathcal{HG})$. Using the verification Theorem 7.3 we can sort out from $\mathcal{M}_{k,r}$ the subset $\mathcal{C}_{k,r}(\mathcal{HG})$ of all (k, r)-safe p/t-nets of $\mathcal{M}_{k,r}(\mathcal{HG})$ whose causal behavior includes $\mathcal{L}_{PO}(\mathcal{HG})$. By using Corollary 7.9 we can partially order $\mathcal{C}_{k,r}(\mathcal{HG})$ with relation to inclusion of their causal behavior. The nets with minimal causal behavior with relation to $\mathcal{L}_{PO}(\mathcal{HG})$ are the minimal elements of this ordering.

9 Final Comments and Future Directions

For a matter of convenience we considered in this paper only slice languages that give rise to partial orders with a unique maximal vertex. However there is nothing in principle that prevents us from considering slice languages that represent families of partial orders with arbitrarily many maximal vertices as depicted in figure 5. In this sense, slice languages are able to represent families of partial orders which are not included in the behavior of any bounded p/t-net. It

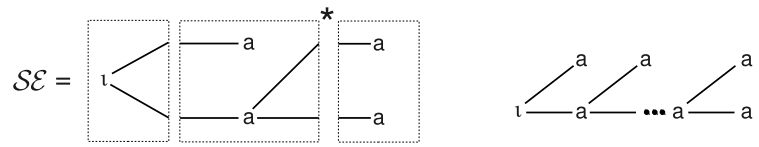

Fig. 5. A slice expression $\mathcal{SE}$ and an intuitive representation of its partial order language. This language cannot be a subset of the behavior of any bounded p/t-net, since it contains partial orders of arbitrarily large width.

would be interesting to characterize which classes of unbounded behaviors one could specify by means of slice languages.

Most of our results were stated in terms of Hasse diagram generators. It turns out that for a matter of flexibility of specification, it might be convenient to allow in the graph language of a slice graph the presence of DAGs which are not transitive reduced. Edges not belonging to the transitive reduction of such DAGs, could be used to highlight particular causal dependences between some of their events. However, Hasse diagram generators are more amenable to formal analysis, since graph languages and partial order languages are in one to one correspondence. Thus it would be convenient to develop a method to transform an arbitrary slice graph into a Hasse diagram generator with identical partial order language. This problem, which is formalized below, is topic of our current research.

Problem 9.1. Given an arbitrary slice graph $\mathcal{SG}$ compute a Hasse diagram generator $\mathcal{HG}$ with $\mathcal{L}_{PO}(\mathcal{SG}) = \mathcal{L}_{PO}(\mathcal{HG})$.

Any bounded p/t-net can be transformed into a one-safe labeled p/t-net with the same causal (and hence execution) behavior [7]. It would be interesting to reverse the direction of this process, i.e. given a one safe labeled p/t-net N_O we would be interested in synthesizing an unlabeled p/t-net with identical partial order behavior, if it exists. By an adaptation of Theorem 7.8, this problem can be reduced to the synthesis of p/t-nets from slice graphs. Indeed from N_O, we derive its Hasse diagram generator $\mathcal{HG}_{sem}(N_O)$ ignoring the labels of the transitions. Subsequently, in each of its slices, we relabel the center vertices with the labels of the transitions. Then we ask for the synthesis of an unlabeled net N whose causal or execution behavior minimally includes the set of partial orders generated by $\mathcal{HG}_{sem}(N_O)$. The advantage of this approach is that $\mathcal{L}_{PO}(\mathcal{HG}_{cau}(N_O))$ is prefix closed and $\mathcal{L}_{PO}(\mathcal{HG}_{ex}(N_O))$ is both prefix and sequentialization closed. These closure conditions are essential if we are aiming to test whether the partial order language specified by a given Hasse diagram generator precisely matches the partial order language of the net synthesized from it.

References

1. E. Badouel and P. Darondeau. On the synthesis of general Petri nets. Technical Report PI-1061, IRISA, 1996.
2. E. Badouel and P. Darondeau. Theory of regions. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 529–586, 1998.
3. R. Bergenthum, J. Desel, G. Juhás, and R. Lorenz. Can I execute my scenario in your net? viptool tells you! In *Proc. of ICATPN 2006*, volume 4024 of *LNCS*, pages 381–390, 2006.
4. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from finite partial languages. *Fundamenta Informaticae*, 88(4):437–468, 2008.
5. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from infinite partial languages. In *Proc. of 8th ACSD*, pages 170–179. IEEE, 2008.
6. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from scenarios with viptool. In *Proc. of 29th ICATPN*, volume 5062 of *LNCS*, pages 388–398, 2008.

[6] For the execution semantics.

7. E. Best and H. Wimmel. Reducing k -safe Petri nets to pomset-equivalent 1-safe Petri nets. In *Proc. of 21th ICATPN*, volume 1825 of *LNCS*, pages 63–82, 2000.
8. P. Darondeau. Deriving unbounded Petri nets from formal languages. *LNCS*, 1466:533–548, 1998.
9. P. Darondeau. Region based synthesis of P/T-nets and its potential applications. In *Proc. of 21th ICATPN*, volume 1825 of *LNCS*, pages 16–23, 2000.
10. R. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.
11. A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. Part I: Basic notions and the representation problem. *Acta Informatica*, 27(4):315–342, 1989.
12. A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. Part II: State spaces of concurrent systems. *Acta Informatica*, 27(4):343–368, 1989.
13. J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the EATCS*, 52:244–262, 1994.
14. U. Goltz and W. Reisig. Processes of place/transition-nets. In *Proc. of ICALP*, volume 154 of *LNCS*, pages 264–277, 1983.
15. P. Hoogers, H. Kleijn, and P. Thiagarajan. A trace semantics for Petri nets. *Information and Computation*, 117(1):98–114, 1995.
16. L. Jategaonkar and A. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154:107–143, 1996.
17. G. Juhás, R. Lorenz, and J. Desel. Can I execute my scenario in your net? In *Proc. of 26th ICATPN*, volume 3536 of *LNCS*, pages 289–308, 2005.
18. R. Lorenz and G. Juhás. Towards synthesis of Petri nets from scenarios. In *Proc. of 27th ICATPN*, volume 4024 of *LNCS*, pages 302–321, 2006.
19. E. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.
20. W. Vogler. Petri nets and their semantics. In *Modular Constructions and Partial Order Semantics of Petri Nets*, volume 625 of *LNCS*, pages 9–32, 1992.

10 Appendix

Proof of Proposition 2.2 Let $M = (Q, \Sigma, Q_0, Q_F, \tau)$ be a nondeterministic finite automaton generating $\mathcal{L}$ where Q is a set of states, $Q_0 \subseteq Q$ a set of initial states, $Q_F \subseteq Q$ a set of final states and $\tau \subseteq Q \times \Sigma \times Q$ the transition relation of M. We write simply qaq' for an element $(q, a, q') \in \tau$. Then $\mathcal{L} = \mathcal{L}(G)$ for a graph $G = (V, E, l)$ such that $V = \tau$, $l(qaq') = a$ for $qaq' \in V$, $V_\iota = \{q_0aq \in V | q_0 \in Q_0\}$, $V_\varepsilon = \{qaq_F \in V | q_F \in Q_F\}$ and $(q_1a_1q_1', q_2a_2q_2') \in E$ iff $q_1' = q_2$ and $q_1a_1q_1' \in \tau$ and $q_2a_2q_2' \in \tau$.

Conversely, let $G = (V, E, l)$ be a labeled graph, with $l : V \to \Sigma$, possibly containing loops, with respective initial and final vertex sets V_ι, V_ϵ. First we add to G two new vertices v_ι, v_ε. Then we connect v_ι to each vertex in V_ι and each vertex in V_ε to v_ϵ. We have that $\mathcal{L}(G)$ is equal to the regular language generated by $M = (Q, \Sigma, Q_0, Q_F, \tau)$ where $Q = E$, $Q_0 = \{(v_\iota, v) \in Q\}$, $Q_F = \{(v, v_\varepsilon) \in Q\}$, and $\tau = \{(v, v')a(v', v'') | (v, v'), (v', v'') \in Q, l(v') = a\}$. □