Place Chart Nets and their synthesis *

Michael Kishinevsky, The Univ. of Aizu, Japan
Jordi Cortadella, Univ. Politecnica de Catalunya, Barcelona, Spain
Alex Kondratyev, The Univ. of Aizu, Japan
Luciano Lavagno, Politecnico di Torino, Italy
Alexander Taubin, The Univ. of Aizu, Japan
Alex Yakovlev, Univ. of Newcastle upon Tyne, England

Abstract

*A model, called Place Chart Nets (*PCN*), is presented. It allows the modelling of both asynchronicity and exception handling (preemption). Contrary to State Charts and other reactive models, which are inherently synchronous,* PCN*s specify a system behavior using partial orders. Contrary to Petri nets,* PCN*s have a notion of hierarchy. Contrary to other hierarchical models based on Petri net extensions, the hierarchy in* PCN*s is determined by preemption. We show that* PCN*s are a non-trivial generalization of classical* PN*s, in the sense that (1) for the finite (bounded) case modeling a* PCN *may require an exponentially larger* PN*, (2) for the infinite (unbounded) case a class of* PCN *languages properly includes a class of* PN *languages and (3) coverability of a marking and k-boundedness of a submarking for* PCN*s are decidable. Then we present a method for synthesis of safe* PCN*s from transition systems, which generalizes the theory of regions previously developed for* PN*s and extend the results for unsafe (but bounded)* PCN*s and for systems with priorities.*

1 Introduction

The motivation for this work lies in a key paper by Drusinsky and Harel ([15]), showing that three, orthogonal modeling paradigms can be used to reduce the size of finite automata-based specifications. By judiciously combining non-determinism, concurrency and hierarchy in a single, graphical finite-state specification model, Harel developed StateCharts in [22], one of the most successful models for hardware and software specification [1].

StateCharts are inherently a *synchronous* model, because all concurrent components share a single notion of *time* and of each other's *state*. The synchronous hypothesis, that also underlies a family of textual languages with similar properties ([8, 21]), is unfortunately difficult to implement for *heterogeneous* architectures, combining software executing on multiple processors with specialized hardware components. Such heterogeneous architectures are essential to design modern embedded controllers, that are being used for a variety of applications, ranging from multi-media terminals to cellular phones, to micro-wave ovens, to anti-lock braking systems.

Synchronicity is also well-suited only for *reactive* systems, in which the computation to be performed in response to an input from the environment is relatively simple. In that case, the abstraction that every

*TR 96-2-003, The University of Aizu. Revised version.

[1] The exact claim in [15] was that "and" type non-determinism (in which all non-deterministic choices must be successful), rather than hierarchical states, was the third source of exponential reduction together with "or" type non-determinism and bounded concurrency. Hierarchical states, on the other hand, were shown in that paper to be able to simulate "and" non-determinism with only a polynomial increase in size.

component of the system makes exactly one transition per "instant", and there is a *total ordering* between operations, is a reasonable model of reality. This is not suitable for systems in which the structure of the computation matters more than the structure of the reaction. In data-intensive systems, for example, a *partial order* is often a better specification model than a total order ([27, 11]). A partial order specification allows one to play, among other things, with the scheduling of operations, to achieve the best utilization of a limited set of resources, or to trade off between the amount of buffering and the complexity of control ([10]).

Asynchronicity is a key property of Petri nets, that also explicitly model concurrency and (in their labeled variety) non-determinism. The original Petri net model, unfortunately, lacks a notion of hierarchy. This makes it very difficult, for example, to model *exception* and *error handling*, as well as different *modes of operation* that are so common in embedded control.

The purpose of this work is two-fold. On the one hand, we define a mechanism for introducing hierarchy into Petri nets, called *Place Chart nets*, and we show that they can be used to achieve an exponential reduction in size with respect to "classical" labeled Petri nets. Although quite a few models have been presented for introducing hierarchy in PNs (see, e.g., [6, 24, 16, 35]) and at least one of them attempted to couple StateCharts with PNs [24]), to the best of our knowledge, the PCN model for the first time provides a concise and consistent coupling of preemption and PNs in such a way that the hierarchy in the model is completely determined by preemption. On the other hand, we show how a PCN can be synthesized from a *flat*, state-based representation, using a notion that is closely patterned on that of *regions* for Petri nets ([18, 31]).

The main significance of this work, in our opinion, lies in this second aspect, that opens promising avenues for *synthesis* and *optimization* of mixed control- and data-flow applications. The control structure and the data dependencies can be represented *uniformly* by using concurrency, causality, choice, suspension and pre-emption[2]. It should be clear to anybody familiar with Petri nets that pre-emption, that is emptying a subnet of all the tokens it contains upon the firing of an event, cannot be represented easily by PNs. We show in Section 4 a construction, based on a rather general technique for PN synthesis, that is exponential in the worst case, and we conjecture that this bound is tight.

Consider, as a motivating example, the following case. The receiving end of a transmission link must perform a CRC check on each message, as well as analyze the message header to determine its handling by the following stages. Those operations can be performed in parallel, but the final handling can be committed only if the CRC check is successful. Suppose, for the sake of simplicity, that the required header handling is to increment a "hop" counter in the header, and look up a Content Addressable Memory to extract the next destination.

The behavior can be specified initially by using three separate "modules". A synchronous specification, e.g. with StateCharts or Esterel would require to fully describe the relative scheduling of the steps executed by each module. For example, it would be necessary to specify if the CRC computation is performed in the same amount of time as the routing decision. Even when concurrency is used in these languages, the synchronous hypothesis determines a priori which steps are executed simultaneously, and would require for example to decide if computeCRC takes one or two time slices (i.e., the same time as getHDR or getHDR plus readCAM). On the other hand, an asynchronous specification has the flexibility to determine the actual scheduling only at synthesis time, when it will be known, for example, if the CRC check is executed in hardware, one bit at a time, or in software, one word at a time.

A Petri net representation of the given specification is shown in Figure 1. Assume that the header processing is an expensive operation, and that it uses some resources that can also be used by a lower-priority operation (e.g., a maintenance operation to modify the routing tables). In that case, one can achieve better average-case performance by *aborting* its operation if the CRC check fails, without waiting for its completion. Petri nets do not offer good facilities for such "abortion" operations, because a transition

[2]Informally, by suspension of some activity by an event we mean that the event interrupts the activity, and that the activity will be resumed later on from the same state. In case of pre-emption, on the other hand, the interrupted activity is resumed (if at all) from its starting state, regardless of the state where it was interrupted. Suspension can be modeled in PNs by removing a token from a self-loop place connected to all transitions of the suspended activity.

can fire only if *all* its input places are enabled. Removing the tokens that are traveling inside a sub-network requires a number of transitions that is proportional to the number of reachable markings of that sub-network (i.e., a potentially exponential number).

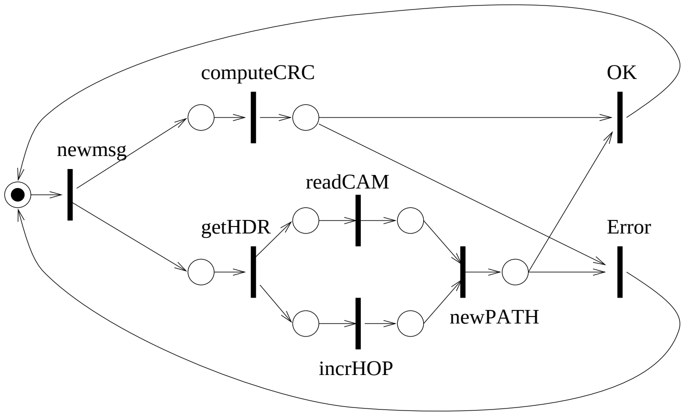
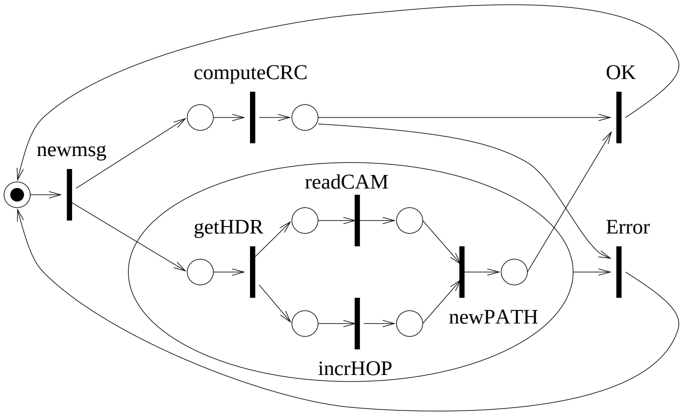


Figure 1: A PN model of a transmission protocol fragment

On the other hand, a PCN shown Figure 2 offers a more compact representation, with a hierarchical StateChart-like flavor, of the same specification *including abortion in case of error*. In this representation a firing of transition `Error` is enabled when a token is present in *any one* of the places of the header processing subnetwork, and removes *all* tokens from it. The representation is handy, because it allows several alternative implementations, without requiring full scheduling, as in the synchronous StateChart case, nor a complete enumeration of the "pre-empted" markings, as in the Petri net case. One example is an exception-based implementation in software, in which the handler of the exception raised by the CRC checker (which could be implemented in hardware) simply restarts the header processing thread. Another example is a polling-based implementation, in which the status of the CRC is examined at some specific checkpoints.



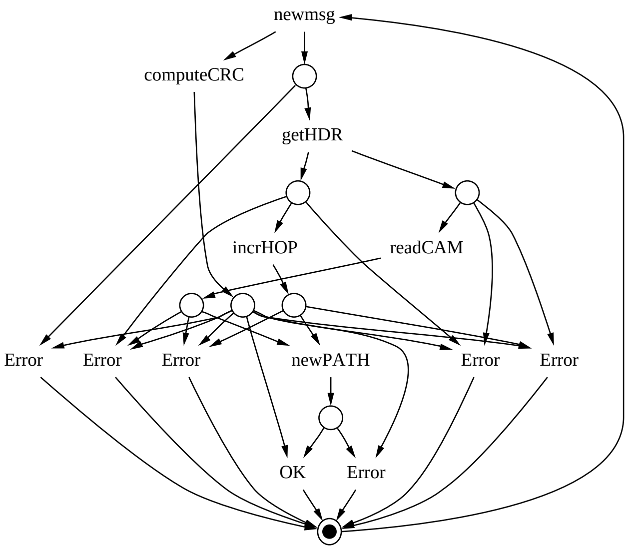


Figure 2: A PCN model of a transmission protocol fragment (to the left) and a "flat" PN behaviourally equivalent to the PCN model (to the right)

To illustrate the gain in descriptive power resulting from the use of PCN, we present a behaviourally equivalent PN model in Figure 2. The latter was obtained by means of, firstly, constructing the reachability graph for the PCN and, secondly, synthesising a safe PN from it using the method described in [12, 13] (both steps were done automatically using tool petrify).Such a"flat" PN has six copies of the transition labelled `Error`. Each copy stands for a separate marking of the header-handling sub-network.

The PCN representation in Figure 2 is also interesting because one could think of initially specifying the set of system states in which pre-emption could take place by using any syntactic mechanism, including explicit state enumeration, and then identify a *maximal sub-network* that structurally corresponds to that set of markings. That sub-network could then become the basis for a syntax-directed implementation in software or hardware of the given behavior. This optimization task is fairly general in nature, and amounts to efficiently implementing a *control* strategy on top of a *data flow* algorithm.

This paper is organized as follows. Section 2 defines Place Chart nets, their syntax, action rules, hierarchy, main classes and operational semantics. It also formally defines Transition System and their behavioural equivalence based on bisimulation. Section 3 presents two characteristic examples of preemption and self-preemption in PCNs. Section 4 analyzes the descriptive power of PCNs for finite (modelling of PCNs with PNs) and infinite (language class containment) cases, and discusses decidability of coverability and k-boundedness problems. Section 5 describes a region-based theory for the synthesis of safe PCNs from transition systems and then extends the synthesis procedure for bounded PCNs. Section 6 extends the results for systems with priorities. Section 7 draws conclusions. Appendix I contains the proofs of the main statements. Appendix II shows a method for flattening PCNs based on insertion of silent transitions.

2 Place Chart Nets

A Place Chart Net (PCN) is a quadruple $N = (C, T, F, m_0)$. $C = \{c_1, \dots, c_m\}$ is a *prefix-closed* set of words of length not greater than k over a *finite* alphabet $P = \{p_1, \dots, p_n\}$. Alphabet P is called a place alphabet. Set C is called a set of place charts. Parameter k is called the depth of the hierarchy. T is a *finite* set of transitions, $F : (C \times T) \cup (T \times C) \longrightarrow \mathbb{N}$ is the flow function (also called the weight function) and $m_0 : C \longrightarrow \mathbb{N}$ is a function called the initial marking. A precise definition of the marking function and some restrictions on the flow function will be discussed in the following. In particular, isolated place charts are not allowed, i.e.,

$$\forall c \in C : {}^\bullet c \cup c^\bullet \neq \emptyset$$

This requirement implies the following property: no redundant levels of hierarchy are present in any PCN specification. Consider, e.g., a place chart $c \in C$. Each non-empty prefix of c represents a non-isolated place chart, i.e., if $c < b$ (and hence b is a prefix of c), then ${}^\bullet b \cup b^\bullet \neq \emptyset$.

If $F(c,t) > 0$ we say that c is an input place chart for t and t is an output transition for c. Similarly, if $F(t,c) > 0$ then t is said to be an input transition for c and c is an output place chart for t. The sets of input and output place charts of transition t are denoted by ${}^\bullet t$ and $t^\bullet$. The sets of input and output transitions of place chart c are denoted by ${}^\bullet c$ and $c^\bullet$.

Place Charts hierarchy. The definition of place charts implies a hierarchy based on place inheritance. It can be formally established using the following definitions.

- A place chart c_i is a *child* of c_j (and c_j is a *parent* of c_i) if $\exists p_l \in P : c_i = c_j p_l$ [3].
- A place chart c_i is a *descendant* of c_j (and c_j is an *ancestor* of c_i) if there is a nonempty word s such that $c_i = c_j s$. Obviously, $|c_j| + |s| \leq k$.
- A place chart is called a *top place chart* if it has no parents.
- A place chart is called a *bottom place chart* if it has no children.

[3] Note that the definition of a chart set allows only single inheritance of place charts, since each prefix of a place chart c is uniquely defined.

- A place chart is called a *place* if it has no children and no parents (an orphan in both directions). A simple place (analogous to a PN place) is simultaneously a bottom and a top place chart.

We will use symbols $<<$ and $<$ to denote the child relation and the descendant relation, respectively. I.e, we write $b << c$ if b is a child of c and $b < c$ if b is a descendant of c. Furthermore, we will use notation $b \leq c$ if $b = c$ or $b < c$. Thus, $<$ is the transitive and $\leq$ is both transitive and reflexive (but antisymmetric) closure of $<<$. The set of children of a place chart net is denoted by $C(c) = \{c' : c' << c\}$.

A place chart is called *trivial* if it has only one child. Trivial place charts can be sometimes useful for modeling systems with preemption: they provide a mechanism for removing all tokens from a simple place.

The set of place charts C together with the $<<$ relation can be represented as a tree. The root of this tree is an isolated place chart c^{ϵ}, which corresponds to the empty word ϵ in C. We define the notion of scope for a place chart and a transition. The *scope of a place chart* (other than c^{ϵ}) c is simply its parent. The *scope of a transition* t, denoted as $scope(t)$ is a place chart c such that c is the *least common ancestor*, with respect to relation $<$ for the set $^{\bullet}t \cup t^{\bullet}$. Note that using $<$ instead of $\leq$ ensures that $c \notin {}^{\bullet}t \cup t^{\bullet}$.

Place Charts notation. To denote a place chart we will use a dot notation, like $p_1.p_2.p_1$. To denote sets of place charts it is convenient to factor parents out of the children. E.g., a set of place charts: $\{p_1.p_2.p_1, p_1.p_2.p_2, p_1.p_2.p_3, p_1.p_3, p_2.p_3\}$ can be also denoted as $\{p_1(p_2(p_1, p_2, p_3), p_3), p_2.p_3\}$.

Initial place charts. The flow relation of a *well-formed* PCN is restricted to have only *bottom place charts* as sinks, i.e., $\forall t \in T, c \in C : F(t, c) > 0 \Rightarrow \neg(\exists p \in P : cp \in C)$.

In order to make the notation more compact in the following (and in order to pay homage to the original StateChart syntax) we will also use a shorthand notation that “factors” out common incoming edges. Let c, c' be two place charts such that c is bottom and $c << c'$. Given an integer n and a set T' of transitions such that $F(t, c) \geq n$, we can also *equivalently* use a different flow relation F' such that $F'(t, c') = n$, $F'(t, c) = F(t, c) - n$ and $c \in In(c')$. We call $In(c')$ the set of *initial place charts* for c'. Initial place charts are denoted with a dangling input arc in the figures. Note that this factorization is not unique in general.

Marking. A marking is a function $m : C \longrightarrow \mathbb{N}$ which is recursively defined as follows

1. For each bottom place chart c, $m(c)$ is a natural number.

2. If c is a place chart, then
$$m(c) = max_{c' \in C(c)} \{m(c')\}$$
That is, the marking of a place chart is the maximum of the markings of its children and hence a marking of a PCN is completely defined by the marking of its bottom place charts. In the sequel we refer to this definition of the marking function as the *max rule.*

An initial marking is given as a part of the initial specification. Markings evolve following two rules: the enabling rule and the firing rule.

Enabling rule. It is exactly the same as in PNs: a transition $t \in T$ is enabled at marking m if each input place chart c is marked with at least $F(c, t)$ tokens. More formally:

$$\forall c \in {}^{\bullet}t : m(c) \geq F(c, t)$$

Firing rule. A transition t enabled at marking m_1 may fire, producing a new marking m_2 (denoted $m_1 \xrightarrow{t} m_2$) following two rules: the decrementing rule and the incrementing rule.

- *Decrementing rule:* [4] For any $c \in {}^{\bullet}t$:

[4] Note that in this paper we allow only complete preemption of place charts: if preemption occurs, then all preempted place charts are reset. It is possible to define a similar model in which all places are decremented by a constant amount, regardless of their level in the hierarchy.

- if c is a bottom place chart and there are no ancestors of c in ${}^\bullet t$, then $m_2(c) = m_1(c) - F(c,t)$
- if c is not a bottom place chart (has some children), then for every $b \leq c : m_2(b) = 0$. In other words, for all descendants of c and for the c itself markings are emptied.
- apply the max rule to all ancestors of c.

Note that if both a bottom place and its ancestor are included in ${}^\bullet t$, then the marking of this bottom place is emptied according to the second item of the definition.

- *Incrementing rule:* For any $c \in t^\bullet$: $m_2(c) = m_1(c) + F(t,c)$. Apply the max rule to all ancestors of c.

Reachability. The set of all markings reachable in N from the initial marking m_0 is called its Reachability Set. The *Reachability Graph* (RG) of a PCN is a transition system in which the set of states is the Reachability Set, the events are the transitions of the net and a transition (m_1, t, m_2) exists if and only if $m_1 \xrightarrow{t} m_2$. The relation of reachability on transitions can be trivially extended (by means of the transitive and reflexive closure) to the one on sequences: $m_1 \overset{\sigma}{\Rightarrow} m_2$ if $\sigma \in T^*$ and there is a path in the RG from m_1 to m_2 labelled with transitions in σ in the same order.

Preemptive and self-preemptive transitions. Refer back to Figure 2. Here, informally, transition labeled *Error* can preempt any place inside the ellipse place chart. Formally, we say that transition t is *preemptive* for a place chart c' if there is a place chart $c \in C$, such that $c' < c$ (c is an ancestor of c') and $t \in c^\bullet$.

Figure 4.d illustrates a special case of preemption, called self-preemption, when some tokens can be immediately returned back to the preempted places (transition r returns tokens to $p0.p1$ and $p0.p2$). Formally, we say that a transition t preemptive for c is *self-preemptive* if $\exists c' \leq c : c' \in t^\bullet$.

Labeled PCNs. A *labeled* PCN is a PCN with a labeling function $\lambda : T \longrightarrow A$ which puts into correspondence every transition of the net with a symbol (called label) from the alphabet A. If no two transitions have the same label (unique labeling), then each transition in the net can be uniquely identified by its label. In such case we can use the label as the name of the transition.

Subclasses of PCNs. If the depth of hierarchy $k = 1$, then a PCN is a PN. A net is called *ordinary* if the codomain of F is $\{0,1\}$. A PCN is called *binary-preemptive* if for any chart $c \in C$ $\forall t \in T : F(c,t) > 1 \Rightarrow$ (c is a bottom chart) $\wedge$ $\forall c'(c < c' \Rightarrow F(c',t) = 0)$. In binary-preemptive PCNs the effect of the marking of a non-bottom chart on the enabling of an output transition is like in ordinary nets.

A place is called *k-bounded* if no more than k tokens can appear in the place. A net is *k-bounded* if all its places are *k-bounded*. A 1-bounded net is called *safe*. A net is called *bounded* if there is such constant k for which the net is k-bounded. A net is called a *pure* net if for each *bottom* place chart c the following condition is satisfied: ${}^\bullet c \cap c^\bullet = \emptyset$. In other words, in pure PCNs self-loops between a bottom place chart and a transition are not allowed. However, it is still possible to have loops for non-bottom place charts. Such loops are called *preemptive loops*. Preemptive loops, which correspond to self-preemption of place charts, can change the marking of the place chart, while self-loops for the bottom place charts never change the marking of those charts.

Operational semantics for PCNs. In this paper we assume the interleaving semantics for PCNs. A version of the theory based on the true concurrency semantics can be considered as well. Two versions of the interleaving semantics are considered:

- *asynchronous* – exactly like in PNs all transitions are assumed to be uniform. If a few transitions are enabled, then one of them is taken for firing without any pre-defined priorities. This interpretation does not allow one to specify so-called hard reactive systems with strict requirements on the latency for interrupts and exceptions. However, it can be applied for specification of soft reactive systems, asynchronous systems, and resynthesis of PNs and transition systems. This interpretation also provides a basic case for developing the theory of systems with hard interrupts.

- *asynchronous with priorities* – a transition with a greater (with respect to $<$) scope has higher priority, i.e., preemptive transitions are fired first, interrupting all internal activities at the lower levels of hierarchy. This semantics is useful for specifying hard reactive systems.

The major part of this paper is devoted to PCNs with asynchronous semantics. We discuss the extension of the theory for the asynchronous semantics with priorities in Section 6.

2.1 Transition systems and bisimilarity

In Section 5 we will show how to synthesize PCNs from finite transition systems.

A *transition system* (TS) is a quadruple [31] $TS = (S, E, T, s_{in})$, where S is a *non-empty* set of *states*, E is a set of *events*, $T \subseteq S \times E \times S$ is a *transition* relation, and s_{in} is an *initial state*. The elements of T are called the *transitions* of TS and will be often denoted by $s \xrightarrow{e} s'$ instead of (s, e, s'). A transition system is *finite* if S and E are finite. In the sequel, only finite transition systems will be considered. A TS can be represented by an arc-labeled directed graph.

The transitive closure of the transition relation T is called the *reachability relation* between states and is denoted by T^*. In other words, state s' is reachable from state s if there is a (possibly empty) sequence of transitions from T: $\sigma = (s, e_1, s_1), ..., (s_k, e_k, s')$. This is denoted by $s \stackrel{\sigma}{\Rightarrow} s'$ or simply by $s \stackrel{*}{\Rightarrow} s'$, if the sequence is not important. We also write $s \xrightarrow{e}$, $s \stackrel{\sigma}{\Rightarrow}$, $\xrightarrow{e} s$, and $\stackrel{\sigma}{\Rightarrow} s$ if there is a state $s' \in S$ such that $s \xrightarrow{e} s'$, $s \stackrel{\sigma}{\Rightarrow} s'$, $s' \xrightarrow{e} s$, or $s' \stackrel{\sigma}{\Rightarrow} s$, correspondingly. Each state is reachable from itself, since we allow empty sequences in the definition.

Every TS $TS = (S, E, T, s_{in})$ is assumed to satisfy the following axioms:

(A1) Every event has an occurrence: $\forall e \in E : \exists (s, e, s') \in T$;

(A2) Every state is reachable from the initial state: $\forall s \in S : s_{in} \stackrel{*}{\Rightarrow} s$.

We will discuss possible restrictions on using self-loops in TSs in section 5.

Behavior of a TS and a PCN can be compared based on bisimulation of the TS and the reachability graph of the PCN.

Definition 2.1 (Bisimulation [3]) *Let $TS_1 = (S_1, E, T_1, s_{in_1})$ and $TS_2 = (S_2, E, T_2, s_{in_2})$ be two TSs with the same set of events. A* bisimulation *between TS_1 and TS_2 is a binary relation R between S_1 and S_2 such that*

(i a) for every $s_1 \in S_1$, there exists $s_2 \in S_2$ such that $s_1 R s_2$.

(i b) for every $s_2 \in S_2$, there exists $s_1 \in S_1$ such that $s_1 R s_2$.

(ii a) for every $(s_1, e, s_1') \in T_1$ and for every $s_2 \in S_2$ such that $s_1 R s_2$, there exists $(s_2, e, s_2') \in T_2$ such that $s_1' R s_2'$.

(ii b) for every $(s_2, e, s_2') \in T_2$ and for every $s_1 \in S_1$ such that $s_1 R s_2$, there exists $(s_1, e, s_1') \in T_1$ such that $s_1' R s_2'$.

Intuitively, conditions (i a) and (ii a) define a simulation of TS_1 by TS_2. Two TSs are said to be *bisimilar* if they can simulate to each other, i.e. there exists a bisimulation between them. The relation "*is bisimilar to*" is an equivalence relation and partitions all TSs into equivalence classes.

3 Preemption in nets. Two examples

This section has the purpose of building an intuitive understanding of the concepts of preemption and self-preemption in nets. The major concepts which are informally defined in this section will be formally presented in the subsequent sections.

3.1 Example 1: preemption

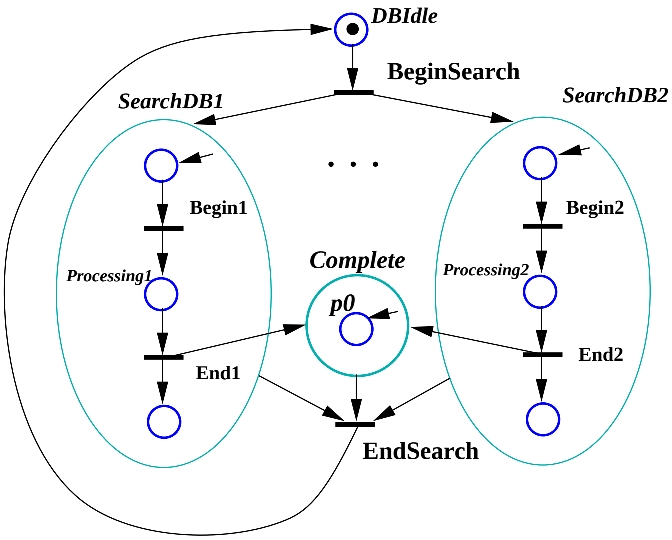


Figure 3: A PCN model of search in distributed database

Consider the PCN model of a search transaction in a distributed database shown in Figure 3. An application process issues a request to the database management system to perform a search of data (transition `BeginSearch`). The system initiates parallel search in several independent database copies (as designated by place charts *SearchDB1* and *SearchDB2*). When one of the subsystems locates the required data (fires its transition `End1` or `End2`), it immediately notifies the overall management system about this by putting a token into place chart *Complete*. The system then aborts all the search processes in the subsystems regardless of which state they are. The effect of aborting is shown by the preempting arcs leading from the place charts to transition `EndSearch`. Note that the rules of action of the PCN allow transition `EndSearch` to remove *all* (again, by means of a pre-emption arc) tokens from place *Complete*, i.e. even when several subsystems have completed their search together. This is achieved by means of using a hierarchical place construct – bottom place chart *Complete.p0* within place chart *Complete*. This place chart does not have any internal events. Its main purpose to define *OR-causality* between the cause-events `End1` and `End2` and the preemptive event `EndSearch`. Note that *Complete.p0* is not structurally isolated, because it has two "factored" input arcs from *End1* and *End2*.

3.2 Example 2: self-preemption

In the previous sections we considered examples of a PCN without chart loops, i.e. after preemption it was not immediately possible to initialize the same chart. Initialization was always mediated via some places/transitions. It is similar to pure PNs. For the purpose of immediate re-initialization of place charts we introduce *self-preemption*. The following example also informally shows the relation between synchronous reactive languages (e.g., Esterel [8]) and PCNs.

Figure 4 shows an example corresponding to the following Esterel program [9]:

loop
  [**await** a || **await** b];
  emit o
each r

The external loop "loop...each r" is a preemptive statement. Every time r occurs, whatever action is executed inside it is stopped, and the loop is restarted from the beginning. Figure 4.a shows a Mealy-like automaton [25] equivalent to this Esterel program. Since in Esterel any emission of an output signal

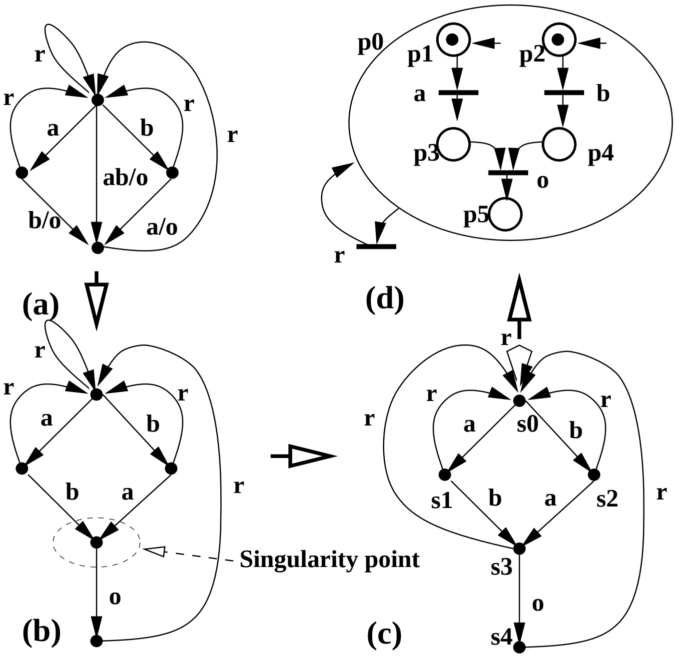


Figure 4: "Mealy-like" Esterel semantics (a), its "Moore-like" TS interpretation (with interleaving) (b), smoothing singularity points (c), corresponding PCN (d)

(o) occurs in 0-time it is not possible to interrupt this emission at the state transition corresponding to multiple transition ab/o. In the Moore machine interpretation this implies that there is a *singularity point* (Figure 4.b), from which there is no preemption. We first remove this singularity (Figure 4.c) since it does not correspond to the event ordering semantics and since it complicates modeling of the specified behavior with PCNs. Then, it is easy to derive the PCN Figure 4.d. Since transition r performs an immediate re-initialization of place chart $p0$ there is a loop between $p0$ and r, which corresponds to self-preemption.

4 Expressive power of PCNs

In this section we show that

- In the finite (bounded) case PCNs can be exponentially more succinct than bisimilar PNs (Section 4.1).
- In the infinite (unbounded) case PCN languages form a proper superclass of PN languages (Section 4.2).
- Contrary to some other extensions of PNs, the k-boundedness property for a place (a set of places) is decidable for PCNs.
- PCNs can exhibit non-commutative behavior.

4.1 Flattening: modeling PCNs with PNs

Flattening is a mechanism for constructing a "flat" PN equivalent (under a certain notion of equivalence) to a hierarchical PCN. Obviously any bounded PCN can be modeled with a PN. We will show in Section 4.2 that this is not the case for unbounded PCNs. However, developing further the argument discussed in the introduction, we will show that any PN equivalent to a certain PCN will be exponentially larger than the PCN. We do not show it for all possible ways of constructing a PN. However, we show it for two rather

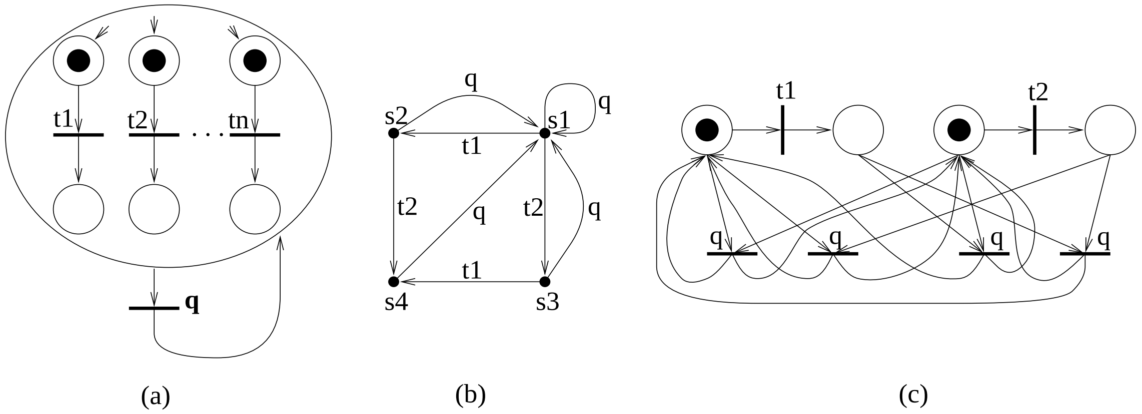


Figure 5: (a) PCN, (b) Reachability graph for $n = 2$, (c) PN for $n = 2$

general techniques. One is based on the theory of region-based PN synthesis, and allows one to build a bisimilar PN by label splitting (renaming of transitions). Moreover, this PN is minimal if no insertion of silent (dummy) transitions is allowed. The other is a structural technique based on inserting silent transitions for each marking in which a preemptive transition is enabled. Both techniques give the same exponential complexity of the modeling PN and we conjecture that this bound is tight. We consider here only the first technique while the structural method is described in Appendix II.

We will use the example in Figure 5 in order to prove the succinctness of PCNs with regard to PNs. Figure 5.a depicts a PCN with n transitions $(t_1, \ldots, t_n)$ representing n concurrent processes. Transition q acts as a reset event that preempts all processes and starts them again. The reachability graph corresponding to such PCN consists of a hypercube of states that represents all possible interleavings of the concurrent transitions. Each state of the hypercube is the source of an arc with label q that goes to the initial state s_1. Figure 5.b shows the reachability graph for two concurrent transitions t_1 and t_2.

Using synthesis techniques based on the theory of regions (see section 5 for details) it is possible to obtain a bisimilar safe PN such as the one depicted in Figure 5.c. In general such PN will have 2^n transitions labeled with event q. This fact can be formally stated as follows:

Theorem 4.1 (Succinctness of PCNs) *Given the* PCN *shown in Figure 5.a, any bisimilar safe* PN *has at least 2^n transitions with label q.*

The proof is presented in the Appendix.

The above theorem proves that PCNs can be exponentially more succinct than PNs to represent some transition systems. In general representing preempting transitions in PNs requires a set of transitions that covers all possible preemptable markings, as illustrated in Figure 5.c.

4.2 Power of PCN languages

In this section we will show that PCN languages constitute a proper (and non-trivial) super-family of the PN languages. We first review some definitions and properties of formal languages.

Definition 4.1 (PNL and PCNL) *[32, 26, 19] Let N be a labeled* PN *or* PCN *with a labeling function $\ell : T \longrightarrow A$ [5]. Then three type of languages can be defined as follows [6]:*

- P-type language (also called prefix language): $L_p(N) = \{\ell(\sigma) \in A^* | \sigma \in T^* \ \wedge \ M_0[\sigma >\}$

[5] ℓ is extended to a mapping $T^* \longrightarrow A^*$ in the usual way.

[6] In [32] T-type languages, $\mathcal{T}$, are also defined as those accepted by terminal marking. Since $\mathcal{T} \subset \mathcal{L}$, they are not of much interest.

- L-type language (marked language): $L_l(N) = \{\ell(\sigma) \in A^* | \sigma \in T^* \wedge M_0[\sigma > M \wedge M \in F\}$ *where* F *is a* finite *set of final markings;*
- G-type language (weak language): $L_g(N) = \{\ell(\sigma) \in A^* | \sigma \in T^* \wedge M_0[\sigma > M \wedge \exists M' \in F : M \geq M'\}$

If the labeling function is λ-free, *i.e., alphabet* A *contains no empty symbol* λ [7], *then the corresponding language is called* λ*-free. Mapping* ℓ *for* λ*-free languages can be viewed as a complete mapping of* T *to* A*, while for non-*λ*-free languages it can be viewed as a partial mapping.*

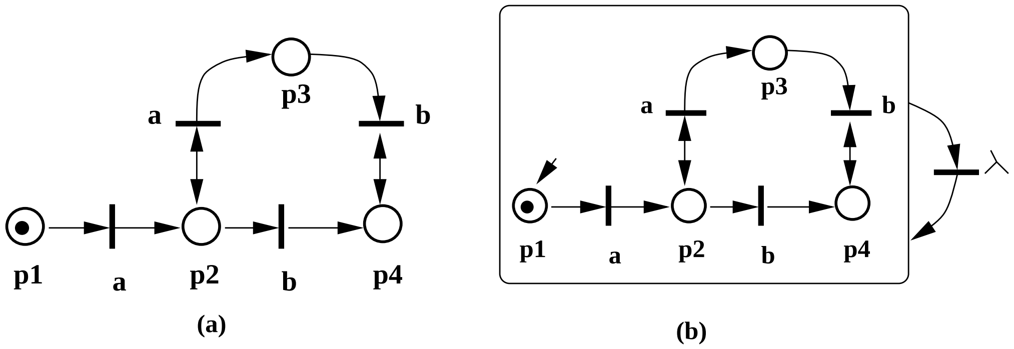


Figure 6: PN generating prefix language $L = \{a^i b^j | i \geq j \geq 0\}$ (a) and PCN generating prefix language $L* = \{(a^i b^j) * | i \geq j \geq 0\}$. $L* \notin \mathcal{L}^\lambda$ (b)

Example. Let N be the PN in Figure 6.a with one final marking $F = \{(0001)\}$. Then $L_l(N) = \{a^i b^i | i > 0\}$, $L_g(N) = \{a^i b^j | i \geq j > 0\}$, and $L_p(N) = \{a^i b^j | i \geq j \geq 0\}$

The classes of P-type, L-type, and G-type languages generated by λ-free labeled PNs (PCNs) are denoted $\mathcal{P}, \mathcal{L}, \mathcal{G}$ ($\mathcal{PC}, \mathcal{LC}, \mathcal{GC}$) respectively. The classes of P-type, L-type, and G-type languages generated by non-λ-free labeled PNs (PCNs) are denoted $\mathcal{P}^\lambda, \mathcal{L}^\lambda, \mathcal{G}^\lambda$ ($\mathcal{PC}^\lambda, \mathcal{LC}^\lambda, \mathcal{GC}^\lambda$) respectively. The following proper inclusions for language classes hold [32, 26]: $\mathcal{P} \subset \mathcal{G} \subset \mathcal{L}$ and $\mathcal{P}^\lambda \subset \mathcal{G}^\lambda \subset \mathcal{L}^\lambda$. Similarly, we can show that $\mathcal{PC} \subset \mathcal{GC} \subset \mathcal{LC}$ and $\mathcal{PC}^\lambda \subset \mathcal{GC}^\lambda \subset \mathcal{LC}^\lambda$. Clearly, since a PN is also a PCN there is an inclusion between the corresponding PNL and PCNL, e.g., $\mathcal{P} \subseteq \mathcal{PC}$, etc. Theorem 4.2 shows that this inclusion is proper.

Let RL, CFL, and CSL stand for regular, context-free and context-sensitive languages. As shown in [32] the following proper inclusions hold: $RL \subset PNL \subset CSL$. Classes PNL and CFL are incomparable. PN languages are closed under concatenation, union, intersection, reshuffling (parallel composition), and reversing. However, they are not closed under the Kleene closure ($*$), complementation and substitution [32]. On the contrary, as follows from the next lemma, P-type PCN languages are closed under the Kleene closure.

Lemma 4.1 *Let* L_p *be a prefix language of a labeled* PN [8] *such that the zero marking is not reachable* [9]*. Then* L_p* *is a prefix language of a non-*λ*-free labeled* PCN *($*$ stands for the Kleene closure).*

The proof is trivial and the idea of the proof is illustrated in Figure 6.b. A PN is encapsulated into one place chart, c, which has one output silent transition. This transition also is an input for c, so it is self-preemptive for any internal place. Since the zero marking is unreachable in N the self-preemptive transition is enabled in every marking and therefore can restart the internal net from the initial marking after any reachable marking.

Obviously if L_p is prefix-closed, then L_p* is also prefix-closed and hence the PCN in Figure 6.b generates a P-type language closed under the Kleene closure. On the other hand, the sub-class of L-type PN languages

[7] The empty symbol corresponds to a silent action.

[8] Labeling can be complete or partial.

[9] This is not a restriction, since any PN can be transformed, without changing its P-type language, to a form when the zero marking is not reachable, e.g., by adding one initially marked place that has self-loops to all transitions of the net.

closed under the Kleene closure coincides with the class of regular languages [32]. By applying the pumping lemma for RL [25] it is easy to show that the P-type language generated by the PCN in Figure 6.b is not regular. This proves the following theorem (a formal proof is given in the Appendix).

Theorem 4.2 *(1) $\mathcal{P}^\lambda$ constitute a proper subclass of $\mathcal{PC}^\lambda$.*
(2) $\mathcal{L}^\lambda$ constitute a proper subclass of $\mathcal{LC}^\lambda$.
(3) $\mathcal{PC}^\lambda$ is not a subclass of $\mathcal{L}^\lambda$.

It is interesting to investigate how far PCNL extend PNL, i.e., what is inside the subclass of languages $\mathcal{PC}^\lambda - \mathcal{P}^\lambda$. As follows from Theorem 4.2 PCN languages include $L*$ for any prefix PN language L and clearly are much wider than the family of prefix-closed regular languages. Moreover as the next lemma, proved in the Appendix, shows there are some languages in $\mathcal{PC}^\lambda - \mathcal{P}^\lambda$ which are not context-free (but are, of course, context-sensitive).

Lemma 4.2 *Some languages in $\mathcal{PC}^\lambda - \mathcal{P}^\lambda$ are not context-free.*

To conclude, PCN languages form a non-trivial extension of PN languages. While PN languages include only finite repetitions of infinite behaviors and infinite repetitions of finite behaviors, PCN languages also include infinite repetitions of infinite (i.e., unbounded) behaviors. The following inclusion holds: $RL \subset PNL \subset PCNL \subset CSL$, while $PCNL$ and CFL are still incomparable.

4.3 Reachability, coverability and k-boundedness of PCNs

PCNs can be modeled with *reset* nets [2, 33, 23] a sub-class of self-modifying nets [2, 33, 34]. Reset nets are flat and have no hierarchy, however they can contain reset arcs. If transition t is connected by a reset arc (p, t) with a place p, then after firing of t all tokens in p are erased. Marking of place p does not have any effect on enabling of transition t[10].

The following properties of reset nets are also valid for PCNs:

- *Coverability of a marking:* Given an integer vector v it is decidable whether there is a reachable marking m such that $m \geq v$. The proof has been given in [4] and is based on building a finite coverability tree for backward reachability in reset nets starting from v[11].
- *k-boundedness:* it follows from decidability of the coverability problem that k-boundedness of a place chart (any subset of place charts) is decidable.
- *Reachability:* [2] shows that it is undecidable whether an arbitrary marking m is reachable in a reset net.

4.4 Commutativity of PCNs

It is known that unlabeled PNs are *commutative* in the following sense: for any reachable marking m and pair of transitions t_1 and t_2 such that $m \overset{<t_1,t_2>}{\Rightarrow} m_1$ and $m \overset{<t_2,t_1>}{\Rightarrow} m_2$ (i.e. both sequences t_1, t_2 and t_2, t_1 are feasible in m) we must have $m_1 = m_2$. For PCNs the situation is different due to the effect of preemption. As Figure 7 shows, PCNs can be non-commutative. This can happen if a transition b, preemptive for some child of a place chart $p2$, is concurrent with another transition a which delivers the token into $p2$. In general, commutativity can be checked by traversing the reachability graph.

[10] Therefore, a reset arc can be also thought as an output arc for t.
[11] For PNs coverability tree is build for forward reachability [28, 32].

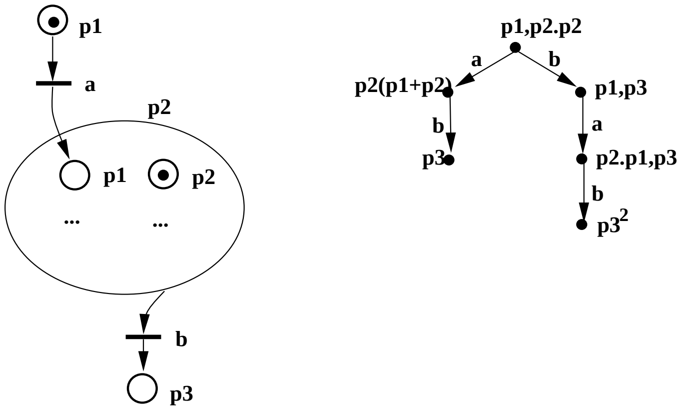


Figure 7: Non-commutativity of PCNs

5 Synthesis of safe PCNs

The theory of regions was first presented in [18, 31] and further applied to PN synthesis in [14, 7, 5, 12, 13]. In this section we generalize the theory of synthesis of safe PNs presented in [12, 13] to synthesis of safe PCNs. Let us consider the following problem:
Given: a transition system TS.
Find: a safe PCN, N_R, which is bisimilar to the TS (under asynchronous operational semantics).

5.1 Region charts

Region charts in a TS are such sets of states which are related to place charts in the corresponding PCN.

Definition 5.1 (Transition predicates) *Let $TS = (S, E, T, s_{in})$ be a* TS. *Let $S' \subseteq S$ be a subset of states and $e \in E$ be an event. The following predicates are defined for S' and e:*

$$\begin{array}{rcl}
\texttt{in}(e,S') & \equiv & \exists (s,e,s') \in T : s, s' \in S' \\
\texttt{out}(e,S') & \equiv & \exists (s,e,s') \in T : s, s' \notin S' \\
\texttt{enter}(e,S') & \equiv & \exists (s,e,s') \in T : s \notin S' \wedge s' \in S' \\
\texttt{exit}(e,S') & \equiv & \exists (s,e,s') \in T : s \in S' \wedge s' \notin S' \\
\texttt{IN}(e,S') & \equiv & \forall (s,e,s') \in T : s, s' \in S' \\
\texttt{OUT}(e,S') & \equiv & \forall (s,e,s') \in T : s, s' \notin S' \\
\texttt{ENTER}(e,S') & \equiv & \forall (s,e,s') \in T : s \notin S' \wedge s' \in S' \\
\texttt{EXIT}(e,S') & \equiv & \forall (s,e,s') \in T : s \in S' \wedge s' \notin S'
\end{array}$$

Definition 5.2 (Region chart) *A set of states $rc \subseteq S$ in $TS = (S, E, T, s_{in})$ is called a* region chart *if* exactly one *of the following three conditions is satisfied for each event $e \in E$:*

(i) Region conditions:

$$(\texttt{enter}(e,rc) \;\Rightarrow\; \neg\texttt{in}(e,rc) \;\wedge\; \neg\texttt{out}(e,rc) \;\wedge\; \neg\texttt{exit}(e,rc)) \;\wedge$$

$$(\texttt{exit}(e,rc) \;\Rightarrow\; \neg\texttt{in}(e,rc) \;\wedge\; \neg\texttt{out}(e,rc) \;\wedge\; \neg\texttt{enter}(e,rc))$$

(ii) Preemption condition:

$$\texttt{exit}(e,rc) \;\wedge\; \texttt{out}(e,rc) \;\wedge\; \neg\texttt{in}(e,rc) \;\wedge\; \neg\texttt{enter}(e,rc)$$

(iii) Self-preemption condition:

$$\texttt{enter}(e, rc) \;\wedge\; \texttt{in}(e, rc) \;\wedge\; \neg\texttt{out}(e, rc) \;\wedge\; \neg\texttt{exit}(e, rc)$$

If the region conditions, (i), are satisfied for all events $e \in E$, then a region chart is called a region.
If the preemption condition, (ii), is satisfied for e, then event e is called a preemptive event *for region chart rc.*
If the self-preemption condition, (iii), is satisfied for e, then e is called a self-preemptive *event for region chart rc.*

Given a region chart rc: rc° is the set of events preemptive for rc, and ${}^\circ rc^\circ$ is the set of events self-preemptive for rc. Finally, let $rc^{\bigcirc} = rc^\circ \cup {}^\circ rc^\circ$.

Given an event e: ${}^\bullet e$ is the set of pre-region charts (shortly pre-regions), i.e., such charts rc that $\texttt{EXIT}(e, rc)$;
$e^\bullet$ is the set of post-region charts (shortly post-regions), i.e., such charts rc that $\texttt{ENTER}(e, rc)$;
${}^\circ e$ is the set of region charts preempted by e, i.e., charts, rc, for which the preemptive condition *(ii)* is satisfied with respect to e;
and ${}^\circ e^\circ$ stands for the set of region charts self-preempted by e, i.e., charts, rc, for which the self-preemptive condition *(iii)* is satisfied with respect to e. Finally, let ${}^{\bigcirc} e = {}^\circ e \cup {}^\circ e^\circ$.

5.2 Region chart hierarchy

The concept of region chart hierarchy is based on the "justified by" relation between region charts. It corresponds to the descendant relation on place charts and is therefore crucial for the synthesis of PCNs.

Definition 5.3 (Full justification) *A region chart rc_1 is said to be* justified *by another region chart rc_2 (denoted: $rc_1 \prec rc_2$) if the following three conditions are satisfied:*

1. $rc_1 \subset rc_2$
2. $rc_1^{\bigcirc} - rc_2^{\bigcirc} \neq \emptyset$ *and* $\forall e \in (rc_1^{\bigcirc} - rc_2^{\bigcirc}) : \texttt{EXIT}(e, rc_2) \;\vee\; \texttt{IN}(e, rc_2)$.
3. $\forall e \in (rc_2^{\bigcirc} - rc_1^{\bigcirc}) : \texttt{ENTER}(e, rc_1) \;\vee\; \texttt{OUT}(e, rc_1)$.

If $rc_1 \prec rc_2$ and there is no other region chart rc_3 such that $rc_1 \prec rc_3 \prec rc_2$, then we write $rc_1 \prec\!\prec rc_2$.
We say that rc is justified if there is a region chart, rc', that justifies rc.
A region chart rc is said to be fully justified *if there is a region r that justifies rc.*

The partial order $\prec$ corresponds to the "descendant-ancestor" relation for place charts, while $\prec\!\prec$ corresponds to the "child-parent" relation.

Definition 5.4 (Relative complement) *If $rc' \prec rc$, then $rc - rc'$ is called the* relative complement *of rc' with respect to rc.*

The following remarks help to clarify the definition of justification.

- According to the second condition of Definition 5.3, at least one transition which was preemptive or self-preemptive for rc_1 should become *an exit or an internal* transition for rc_2. These are the only two forms of legal conversion of preemption into a "normal" transition which does not violate region conditions. Additional intuition behind this restriction is given by Lemma 5.6: if this condition is violated, then the complement of a child region chart with respect to its parent region chart is not necessarily a region chart.

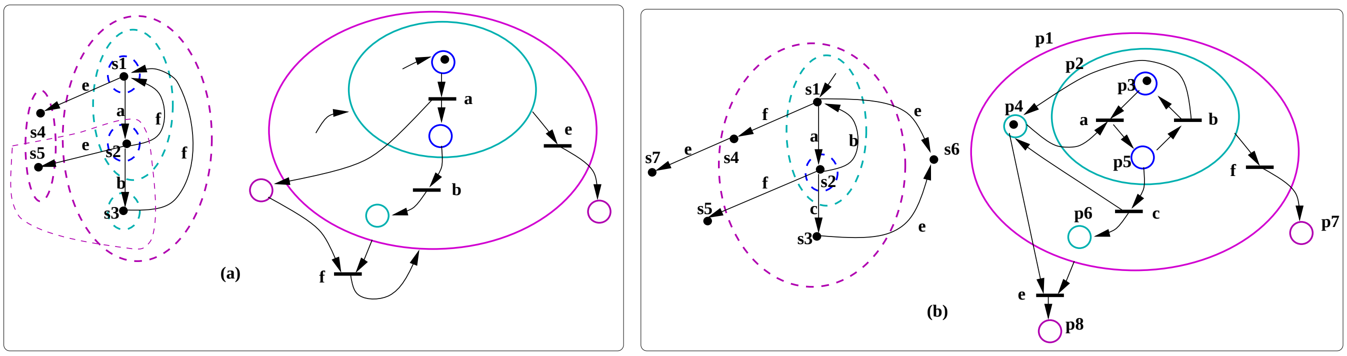


Figure 8: Parent with a new self-preemptive (a) and preemptive (b) event

- According to the third condition of Definition 5.3, a new preemptive or self-preemptive event e may occur for the parent, but only if condition $\texttt{ENTER}(e, rc_1) \ \vee \ \texttt{OUT}(e, rc_1)$ is satisfied for the child. Figure 8 illustrates why this can happen.

  In Figure 8,a there is a TS and the corresponding PCN. This figure illustrates how a new self-preemptive event may occur in a parent region chart. Consider the following region charts: $rc_1 = \{s_1\}, rc_2 = \{s_2\}, rc_3 = \{s_3\}, rc_4 = \{s_1, s_2\}, rc_5 = \{s_1, s_2, s_3\}, rc_6 = \{s_2, s_3, s_5\}$. Region chart rc_1 is justified by rc_4: the only preemptive event for rc_1 is e, which exits rc_4. However, for rc_4 a new self-preemptive event f appears. Note that $\texttt{ENTER}(f, rc_1)$ is satisfied for the child chart. This self-preemption is cured by rc_5 for which f is internal.

  Figure 8,b illustrates how new preemptive events can occur. We name all place charts with unique names for simplicity. They correspond to the following regions:
  $rc_1 = \{s_1, s_2, s_3, s_4\}, rc_2 = \{s_1, s_2\}, rc_3 = \{s_1\}, rc_4 = \{s_1, s_3, s_4\}, rc_5 = \{s_2\}, rc_6 = \{s_3\}, rc_7 = \{s_4, s_5, s_7\}, rc_8 = \{s_6, s_7\}$.

  Region chart rc_5 is justified by rc_2: the only preemptive event for rc_5 is f, which exits rc_2. However, for rc_2 a new preemptive event e appear. Note that $\texttt{OUT}(e, rc_5)$ is satisfied for the child chart rc_5. This preemption is cured by rc_1 which is exited by e.

- Lemma 5.2 will show that although new preemptions or self-preemptions may occur in parents, once cured a preemption can never appear again, i.e., there is a *monotonous progress* in justifying preemptions.

- Note also that conversion of preemption into self-preemption is allowed by Definition 5.3, i.e., event e preemptive for rc_1 can become self-preemptive for rc_2. The opposite transformation: self-preemptive for rc_1 becomes preemptive for rc_2, although not formally forbidden by the definition, can never occur (see the proof of Lemma 5.6 for details).

- There is a nice symmetry in the definition (see the second and the third item). Justification of a preemptive or a self-preemptive event e can occur only if for the parent rc_2: $\texttt{EXIT}(e, rc_2) \vee \texttt{IN}(e, rc_2)$. New preemptive or self-preemptive events for the parent can occur only if for the child rc_1: $\texttt{ENTER}(e, rc_1) \ \vee \ \texttt{OUT}(e, rc_1)$.

Summary of properties. Lemmas that explain properties of region charts are given below. A summary of Lemmas is given in Table 1. Their proofs are presented in the Appendix.

The following lemma clarifies the conditions of Definition 5.3.

Lemma 5.1

1. *If* $rc_1 \prec rc_2 \ \wedge \ e \in {}^\circ rc_1^\circ \ \wedge \ e \notin rc_2^{\bigcirc}$, *then condition* $\texttt{IN}(e, rc_2)$ *is satisfied.*

Lemma	What is it about
5.1	More detailed conditions of region chart justification
5.2	Monotonous progress in removing preemption
5.3	All states are covered by bottom charts
5.4	All states are covered by top charts
5.5	Preemption is removed *both* for a child and its relative complement
5.6	The relative complement of a descendant is also a descendant of the same ancestor
5.7	Preemption of a chart is preemption for any child or its relative complement
5.8	Marking m is bisimilar to state s if bottom places of m and bottom regions covering s correspond

Table 1: Summary of Lemmas.

2. If $rc_1 \prec rc_2 \ \wedge \ e \in rc_2^{\circ} \ \wedge \ e \notin rc_1^{\bigcirc}$, then condition $\mathtt{OUT}(e, rc_1)$ *is satisfied.*

Item (1) shows that self-preemption is always removed by including the event inside the parent. Item (2) shows that only those events which are outside a region chart can become preemptive for the parent chart. The proof follows immediately from Figure 9,b and d.

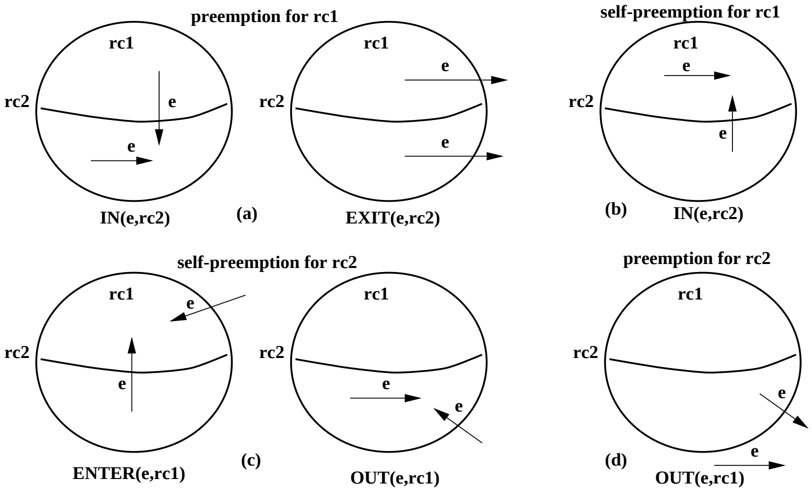


Figure 9: Removing preemption (a) and self-preemption (b) in the parent of a region chart. Appearance of self-preemption (c) and preemption (d) in the parent of a region chart.

The following Lemma proves monotonous progress in removing preemption. Informally speaking, if preemption was removed in the parent of a child, then it cannot appear again in a grandparent.

Lemma 5.2 *Let $rc_1 \prec rc_2 \prec rc_3$. Then*

$$(e \in rc_1^{\bigcirc} \ \wedge \ e \notin rc_2^{\bigcirc}) \Rightarrow e \notin rc_3^{\bigcirc}$$

Since we synthesize PCNs *without multiple inheritance* an additional constraint on region chart justification is required:

Definition 5.5 (Unique justification) *A set of region charts, R, satisfies the* unique justification condition *if*

$$\forall rc_0, rc_1, rc_2 \in R : \ (rc_0 \prec rc_1 \wedge rc_0 \prec rc_2) \Rightarrow (rc_1 \prec rc_2 \vee rc_2 \prec rc_1)$$

This condition guarantees that each region chart has no more than one parent.

Definition 5.6 (Justification condition) *A set of region charts R derived for a TS is called* justified *if every $rc \in R$ is fully and uniquely justified.*

The justification relation of region charts corresponds to the descendant relation on place charts. Let

$$rc_1 \prec\prec rc_2 \prec\prec \ldots \prec\prec rc_i \prec\prec rc_{i+1} \prec\prec \ldots \prec\prec rc_m$$

be such that rc_1 does not justify any other chart and rc_m is a region. Then in the corresponding PCN rc_1 will correspond to a bottom place chart, and $rc_2, \ldots, rc_m$ will correspond to its ancestors, with rc_m corresponding to a top place chart.

Definition 5.7 (Bottom and top region charts) *Region charts which do not justify any other region charts are called* bottom *region charts. Region charts which are not justified by any other region chart are called* top *region charts. We define two predicates on a set of region charts, $bottom(rc)$ and $top(rc)$, which are true if rc is a bottom and top region chart respectively.*

In the example above rc_1 is a bottom region chart and rc_m is a top region chart.

5.3 Self-loops and self-preemption

For synthesis of non-pure PNs with self-loops, irredundant regions which contain all transitions of the "self-loop" event should be considered [12, 13]. Similarly, for synthesis of non-pure PCNs *bottom* region charts, including all transitions of the self-loop event, must be considered.

For constructing the flow function of a *pure* PCN *with self-preemption* it is necessary to consider certain *non-bottom* region charts including all transitions of a self-preemptive event.

Definition 5.8 (Self-loops and preemptive loops)

1. *A region chart rc is called a* self-loop pre-region of event e *if*

$$\mathtt{IN}(e, rc) \ \wedge \ bottom(rc)$$

2. *A region chart rc is called a* preemptive loop pre-region of event e *if e is self-preemptive for one of the children of rc, but is not preemptive for rc, i.e.,*

$$\exists rc_1 : rc_1 \prec\prec rc \ \wedge \ e \in {}^\circ rc_1^\circ \ \wedge \ e \notin rc^{\bigcirc}$$

As follows from Lemma 5.1 if a region chart rc is a preemptive loop pre-region of e, then condition $\mathtt{IN}(e, rc)$ is satisfied.

The set of the self-loop pre-region charts of event e is denoted by ${}^\bullet e^\bullet$. The set of the preemptive loop pre-region charts of event e is denoted by ${}^\star e^\star$. Obviously, since a self-loop pre-region is a bottom region chart, while a preemptive loop pre-region is, by definition, a non-bottom chart, then ${}^\star e^\star \cap {}^\bullet e^\bullet = \emptyset$.

5.4 Excitation regions

While region charts in a TS are related to place charts in the corresponding PCN, the excitation region [30] for event a is a maximal set of states in which transition a is enabled. Therefore, excitation regions are related to transitions of the PCN.

Definition 5.9 (Excitation region) *A set of states S' is called a* generalized excitation region *for event a, denoted by $GER(a)$, if it is the* maximal *set of states such that for every state $s \in S'$ there is a transition $s \xrightarrow{a}$.*

5.5 Justified excitation closed TSs

Depending on the target of the synthesis procedure, i.e. on the type of PCNs to be synthesized, the concrete version of the excitation closure condition can vary. The version given below is aimed at synthesizing safe and pure PCNs.

Definition 5.10 (Excitation closure) *A set of region charts R derived for a* TS $TS = (S, E, T, s_{in})$ *is called* excitation-closed *if the following two conditions are satisfied for R:*

- *Event effectiveness: For each event e:*

$$({}^{\bullet}e \cup {}^{\star}e^{\star}) \cap R \neq \emptyset$$

- *Excitation closure: For each event e:*

$$\bigcap_{rc \in ({}^{\bullet}e \cup {}^{\star}e^{\star}) \cap R} rc = GER(e)$$

Note that preemptive loop pre-regions, ${}^{\star}e^{\star}$, are allowed by the definition of the excitation closure. They are necessary for justification of self-preempted region charts. For example, the region chart $r0 = \{s0, s1, s2, s3, s4\}$ in the TS of Figure 4,c is a preemptive pre-region chart for event r. It is the only pre-region chart that satisfies the excitation closure condition for r.

If a PCN without self-preemption is to be constructed, then preemptive loop pre-regions should be excluded from Definition 5.10 (and thus result in a more complex PCN).

Definition 5.11 (ECTS and JECTS) *A* TS $TS = (S, E, T, s_{in})$ *is called:*

1. excitation-closed *(*ECTS*) if there is a set of region charts R which satisfies the excitation closure condition,*

2. justified excitation-closed *(*JECTS*) if there is a set of region charts R which satisfies the justification condition and the excitation closure condition.*

5.6 Inner closure and irredundant set of region charts

In PCNs every marking is completely defined by bottom places. Hence every state in the corresponding TS that is covered by a non-bottom region chart should be also covered by a child of this chart. We call this property the inner closure.

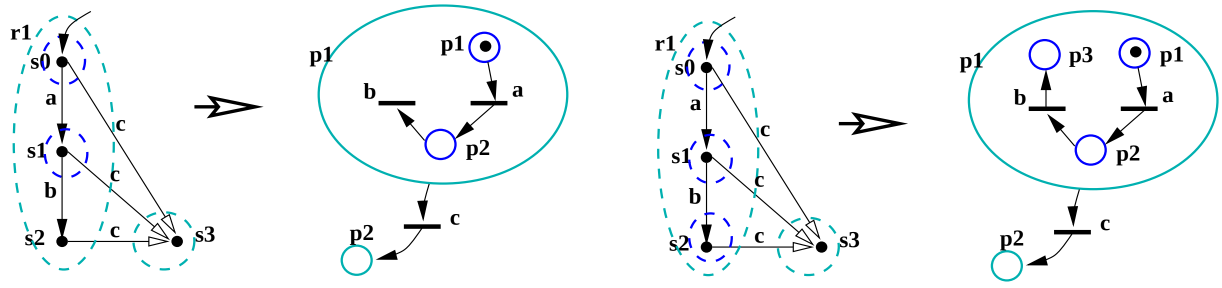
Definition 5.12 (Inner closure) *Let TS be a* TS *with set of states S. A set of region charts R satisfies the* inner closure condition *if*

$$\forall rc \in R\, \forall s \in S\, (\neg bottom(rc) \ \wedge \ s \in rc) \ \Rightarrow \ \exists rc' \in R\, (rc' \prec\!\prec rc \ \wedge \ s \in rc').$$

The inner closure condition guarantees that in the PCN N_R any marked non-bottom place chart can get a zero marking only by preemption. Internal activity of the place chart cannot empty its marking. If the inner closure is not satisfied, then a place chart can be emptied as a result of internal activity. This is illustrated in Figure 10.

Below we give a couple of properties which better explain the structure of region charts. The following lemma shows that if excitation closure and inner closure are satisfied for a set of region charts, then every state of a TS, except possibly a terminal state, is covered by a bottom region chart.

Definition 5.13 (Terminal state) *A state s of a* TS $TS = (S, E, T, s_{in})$ *is called* terminal *if* $\forall e \in E : \neg(s \xrightarrow{e})$.



(a) (b)

Figure 10: The inner closure condition: (a) PCN constructed by a set of region charts violating the inner closure (state $s2$ is not covered by any child of r_1), (b) PCN by a set of region charts satisfying the inner closure

According to our synthesis algorithm, terminal states of a TS can be modeled by a zero marking in the net N_R. For modeling terminal states with non-zero markings (if preferred) every transition of the net N_R without output place charts must be augmented with an output place.

Lemma 5.3 *Let $TS = (S, E, T, s_{in})$ be an* ECTS *and R be a set of region charts for the TS, which satisfies the excitation closure and the inner closure. Let $s \in S$ be a non-terminal state. Then*

$$\exists rc \in R : bottom(rc) \ \wedge \ s \in rc.$$

As a corollary we can show that in a JECTS every state is covered by a top region chart.

Lemma 5.4 *Let $TS = (S, E, T, s_{in})$ be a* JECTS *and R be a set of region charts for the TS, which satisfies the excitation closure, the justification, and the inner closure conditions. Let $s \in S$ be a non-terminal state. Then*

$$\exists rc \in R : top(rc) \ \wedge \ s \in rc.$$

The next Lemma shows that when preemption of a child is removed in a parent, then it is removed both for the child and its relative complement, if the latter is a region chart[12].

Lemma 5.5 *Let rc_1, rc_2 and rc_3 be three region charts, such that $rc_3 = rc_1 \cup rc_2$, $rc_1 \cap rc_2 = \emptyset$, and $rc_1 \prec rc_3$. Then,*

$$e \in rc_1^{\bigcirc} \ \wedge \ e \notin rc_3^{\bigcirc} \Rightarrow e \in rc_2^{\bigcirc}$$

The following lemma shows that the inner closure can be always satisfied, since the relative complement of a descendant region chart to its ancestor is also a descendant region chart of this ancestor.

Lemma 5.6 *Let R be a set of region charts for a* TS. *Let $rc, rc_1 \in R$ be two region charts such that $rc_1 \prec rc$. Then $rc_2 = rc - rc_1$ is a region chart and $rc_2 \prec rc$.*

The next lemma clarifies that e is preemptive (or self-preemptive) for a non-bottom region chart rc only if it is preemptive for any descendant of rc or its relative complement to rc.

Lemma 5.7 *Let rc and rc_1 be two region charts such that $rc_1 \prec rc$. If $e \in rc^{\bigcirc}$, then $e \in rc_1^{\bigcirc} \ \vee \ e \in (rc - rc_1)^{\bigcirc}$.*

Definition 5.14 (Irredundant set of region charts) *Let TS be a* TS. *A set of region charts R is called* redundant *if there is a region chart $rc \in R$ such that $R - \{rc\}$ still satisfies the justification, the inner closure and the excitation closure conditions. Otherwise, R is called* irredundant.

[12] We will show in Lemma 5.6 that this is indeed always the case.

5.7 Synthesis algorithm

The procedure to synthesize a PCN from a JECTS is as follows. Let R be an irredundant set of region charts derived for the JECTS.

Algorithm 1 (TS to safe and pure PCN)

- *For each event e a transition labeled with e is generated in the* PCN*;*
- *For each region chart $rc_i \in R$ a place chart c_i is generated such that the hierarchy of justification on region charts is translated into the hierarchy of place charts in the* PCN*, i.e., if $rc_i \prec\prec rc_{i+1}$, then c_i is a child of c_{i+1}. If $rc_i \in R$ justifies no region charts, then c_i is a bottom place chart.*
- *A bottom place chart c_i contains a token in the initial marking m_0 iff $s_{in} \in rc_i$, where rc_i corresponds to c_i. For other place charts the initial marking is calculated following the max rule.*
- *The flow relation is as follows:*
  - *$F(c_i, e) = 1$ iff rc_i is a pre-region chart of e or a preemptive loop pre-region of e, i.e.,*
  $$rc_i \in {}^\bullet e \cup {}^\star e^\star$$
  - *$F(e, c_i) = 1$ iff rc_i is a bottom region chart and either rc_i is a post-region chart of e or e is self-preemptive for rc_i, i.e.,*
  $$bottom(rc_i) \ \wedge \ rc_i \in e^\bullet \cup {}^\circ e^\circ$$
- *(Optional). Select initial place charts $In(c')$ for each non-bottom place chart c', and modify the flow relation as described in Section 2*

The last step of the algorithm, that does not have a unique solution in general, is used to simplify the flow relation (the cost function can be to minimize the number of place chart crossings).

5.8 Soundness of synthesis

We should compare the initial JECTS, TS, and the RG of the PCN, RG_R, derived according to Algorithm 1 by using an irredundant set of region charts.

This comparison will be based on an equivalence relation between the states of the TS and markings of RG_R. This relation compares the set of region charts that cover state s, denoted by R_s, and the set of places which are contained in the marking m[13].

Algorithm 1 defines a 1-to-1 correspondence (bijection), denoted here ρ, between region charts and place charts. If region chart rc corresponds to a place chart c, then we write $c = \rho(rc)$ or $rc = \rho^{-1}(c)$. If RC is the set of region charts which corresponds to the set of place charts marked in marking m, then we write: $m = \rho(RC)$.

Definition 5.15 (Region-based state equivalence) *Let TS be a* JECTS *and $RG(N_R)$ be the reachability graph of a* PCN *derived from the TS according to Algorithm 1 by using an irredundant set of region charts R. $\mathcal{R}$ is a binary relation defined as follows:*

$$\forall s \in S \forall m \in RG(N_R) : s\mathcal{R}m \overset{def}{\Longleftrightarrow} (m = \rho(R_s) \ \vee \ (terminal(s) \ \wedge \ terminal(m)))$$

Predicate $terminal(s)$ is true when state s is terminal in the TS. Predicate $terminal(m)$ is true if in the net N_R for every e: $\neg(m \xrightarrow{e})$, i.e., m is a terminal marking.

The decrementing rule influences three types of place charts:

[13] This is a generalization of a region-based equivalence proposed for safe PNs in [13]

1. input place charts of e, i.e., $c : c \in {}^{\bullet}e$,

2. the descendants of input place charts: if $c \in {}^{\bullet}e$ and $b < c$, then as a result of applying the decrementing rule $m(b) = 0$. Note that if $b \in e^{\bullet}$ (i.e., e is self-preemptive for b), then after applying the incrementing rule the marking of b and all its ancestors will become 1 again: $\forall d : b \leq d \Rightarrow m_2(d) = 1$.

3. the ancestors of input place charts: let b be a parent of some input place charts of e. If after removing all tokens from ${}^{\bullet}e$ no child place charts of b contain tokens, then by the max rule $m(b) = 0$. Recursively this can remove tokens from further ancestors of input place charts.

To simplify the proof we first show that equivalence of a marking m and a state s can be defined by observing only bottom places from m and bottom region charts from R_s. Let us denote by m^b the set of bottom places marked in m and by R_s^b the set of bottom region charts which contain state s:

$$m^b = \{b \in m : \forall c \in C \neg(c < b)\}$$

$$R_s^b = \{rb \in R_s : \forall rc \in R \neg(rc \prec rb)\}$$

Note that if the "root" place chart c^{ϵ} is ignored, then a marking m can be represented as a forest. Different trees in this forest correspond to sub-markings of place charts which have no relatives (ancestors or descendants) except c^{ϵ} in common. Leaves of the forest correspond to the bottom places marked in m. Similarly, R_s can be represented as a forest of region charts where edges in the trees correspond to the justification relation. The unique justification condition guarantees that different trees representing R_s will not intersect.

Lemma 5.8 *Let TS be a* JECTS *and $RG(N_R)$ be a reachability graph of a* PCN *obtained according to Algorithm 1 by using an irredundant set of region charts R. Let m be a non-terminal marking of the $RG(N_R)$ and s be a non-terminal state of the TS. Then $m\mathcal{R}s$ iff $m^b = \rho(R_s^b)$.*

Note that terminal states should be considered as a special case [14].

Theorem 5.1 *Let $TS = (S, E, T, s_{in})$ be a* JECTS *and N_R is a* PCN *obtained from an irredundant set of region charts, R, according to Algorithm 1. Then N_R is safe, pure and the labeled* RG *of the N_R is bisimilar to the TS.*

The proof is presented in the Appendix. It is worth mentioning that Algorithm 1 synthesizes a PCN with a hierarchy completely determined by preemption. No place chart is included inside a larger place chart if it is not preempted by some output transition of this parent chart. In fact, under certain conditions it is possible, without changing behavior of the net, to include inside a parent chart the children that are never preempted (since when preemption occurs those place charts are always empty, or they will be emptied by the decrementing rule anyway, even without preemption). Hence the algorithm can exploit the trade-off between the size of the modules (place charts) and the level of observability (how many transitions can be observed outside the chart) in view of a syntax-directed implementation of a PCN in software or hardware.

5.9 Synthesis for TS that are not JEC

Algorithm 1 is effective only if the initial TS is justified excitation closed. If the excitation closure or the justification conditions are not satisfied for a given TS, then it is always possible to convert it to an equivalent JECTS by label splitting. This technique can be incorporated into Algorithm 1 and is explained in [12, 13].

[14] A modification of the synthesis algorithm based on switching closure would guarantee that for terminal states and markings condition $m = \rho(R_s)$ also holds. A switching region for e is a maximal set of states such that $\xrightarrow{e} s$. The switching closure is symmetrical to the excitation closure and uses post-regions instead of pre-regions.

Region chart	${}^{\bullet}rc$	$rc^{\bullet}$	rc°
$rc_1 = \{s_0\}$	b, d, f	a	c, e
$rc_2 = \{s_1\}$	a	b	c, e
$rc_3 = \{s_0, s_1\}$	d, f	c	e
$rc_4 = \{s_2\}$	c	d	e
$rc_5 = \{s_0, s_1, s_2\}$	f	e	-
$rc_6 = \{s_3\}$	e	f	-

Table 2: Region charts with input, output, and preemptive events.

5.10 Synthesis of bounded PCNs

A theory for the bounded case is a generalization of the theory presented in Section 5 and [29].

However, there is one important difference. For the unsafe case one needs a notion of a *floating region*. A floating region allows variations in the decrementing slope of preemptive events. Really, no matter how many tokens there are in a place - all of them have to be removed by preemption. Hence for preemptive transitions of a floating region the following condition should be satisfied [15]:

$$FloatEXIT(e, rc) = \forall(s, e, s')(s \in basis(rc) \wedge s' \notin basis(rc)).$$

Moreover, for the internal transitions the slope can vary as well, but it is required that it is not possible to leave rc by e:

$$FloatIN(e, rc) = \forall(s, e, s') : s, s' \in basis(rc).$$

Full justification of region charts can be done not only by regions but also by floating regions.

5.11 Examples

For the self-preemptive example from Figure 4 the trivial region chart $r0$ participates both in the excitation closure condition (for preemptive event r) and in the justification condition (for all children of $r0$).

To see how the hierarchy is extracted let us consider an example in Figure 11. Table 2 lists region charts for the TS on the left. For each region it also shows the set of input, output and preemptive events.

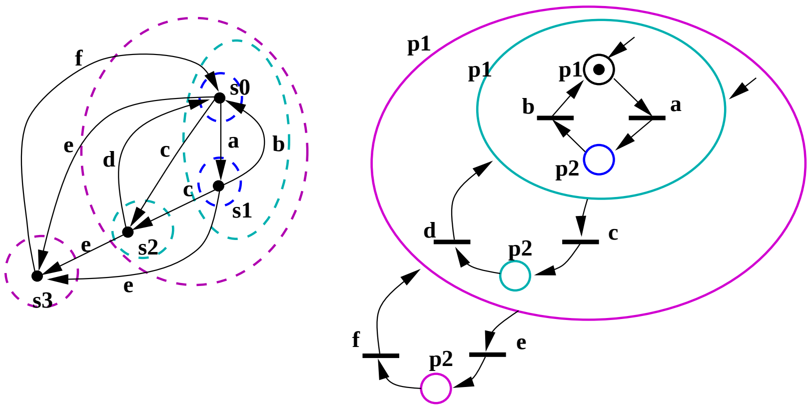


Figure 11: Extracting hierarchy

The excitation closure condition and the unique justification condition are satisfied for this TS. Note that some region charts are used both for excitation closure and for justification. E.g., region chart $rc_3 = \{s_0, s_1\}$

[15] A generalized region for synthesis of bounded nets is a multiset. A *basis* of a multiset is a set which contains all the elements of a multiset without powers. For example, a basis of $r = \{s_1^3, s_2^2, s_3\}$ is a set $\{s_1, s_2, s_3\}$. See [29] for details.

is the only pre-region chart for c and it implies the excitation closure condition for c. However, rc_3 also justifies rc_1 and rc_2. Region chart rc_5 satisfies excitation closure for e and justifies rc_4 and rc_3. Since $rc_0 \prec rc_3 \prec rc_5$, $rc_1 \prec rc_3$, and $rc_4 \prec rc_5$ there are two levels of hierarchy in the corresponding PCN.

6 PCNs with priorities

Modeling priority systems with PNs has been studied both for the interleaving [20] and the true concurrency semantics [17]. In the context of PCNs priorities are especially of interest for modeling exception handling using preemption. For specifying hard reactive systems one needs to assume that preemptive transitions have higher priorities and hence that the internal activity of the module is interrupted as soon as a preemptive transition can fire. We call this vertical priorities (contrary to horizontal priorities between transitions at the same level of hierarchy). This section presents a sketch of the synthesis theory for systems with horizontal and vertical priorities.

6.1 Priority relation for TSs and PNs

Priority relation, $\lhd$, is a binary irreflexive, antisymmetric, and transitive relation on pairs of events. We write $a \lhd b$ if a has less priority than b.

In terms of PNs this means that if both a and b are enabled (ignoring priorities) at marking m, then only b can fire. To be more precise for a PN with priority relation we re-define a notion of enabling:
"e is enabled in $(PN, \lhd)$ under marking m" $=$ e is enabled in PN under marking m and there is no f enabled in PN under m such that $e \lhd f$.

Given a pair $(TS, \lhd)$ we say that the TS is *in agreement with priority* relation $\lhd$, if
$a \lhd b$ implies that there is no state, s, such that $s \xrightarrow{a}$ and $s \xrightarrow{b}$. We will always assume that a TS is in agreement with $\lhd$.

6.2 Synthesis problem for PNs with priorities

A couple of synthesis problems of flat PNs with priorities could be of interest. Let us call them problems for horizontal priorities.

Problem 1 (Flat synthesis constructing priorities):
Given a TS (i.e., a TS without priority relation) build a $(PN, \lhd)$ (i.e., a PN with a priority relation) bisimilar to the TS.

In Problem 1 we synthesize both a PN and a priority relation such that for this priority relation the PN behaves exactly like the TS (as far as the interleaving semantics is concerned).

Problem 2 (Flat synthesis with given priorities):
Given a $(TS, \lhd)$ (i.e., a TS with priority relation) build a $(PN, \lhd)$ (i.e., a PN with the same priority relation) bisimilar to the TS.

Solution for the second problem is in modifying the excitation closure:

$$GER(e) = \bigcap_{rc \in {}^\bullet e} rc - \bigcup_{e \lhd a} GER(a),$$

i.e. from the intersection of preregions of e we need to exclude excitation regions of those events which have higher than e priority.

For the first problem we can use the modified excitation closure condition above under constraints that the derived $\lhd$ must define an antisymmetric partial order relation on the set of events E, i.e., it is never the case that $a \lhd b$ and $b \lhd a$.

Two conditions checked on the TS help to reduce the search for $\lhd$:

- $a \lhd b \ \Rightarrow\ GER(a) \cap GER(b) = \emptyset$

- $a \lhd b \Rightarrow \bigcap_{rc \in {}^\bullet a} rc \cap GER(b) \neq \emptyset$

6.3 Synthesis problem for PCNs with priorities

We call this synthesis problems for vertical priorities.

Here we consider PCNs under asynchronous semantics with priorities. Hence we need to define this semantics precisely.

Definition 6.1 (Dominance relation in PCNs) *Transition b* dominates *transitions a (denoted, $a \unlhd b$) if*

$$\exists c_1, c_2 \in C : c_1 \in {}^\bullet a \ \wedge \ c_2 \in {}^\bullet b \ \wedge \ c_1 < c_2.$$

In other words, one of the input place charts of b is an ancestor of an input place chart of a.

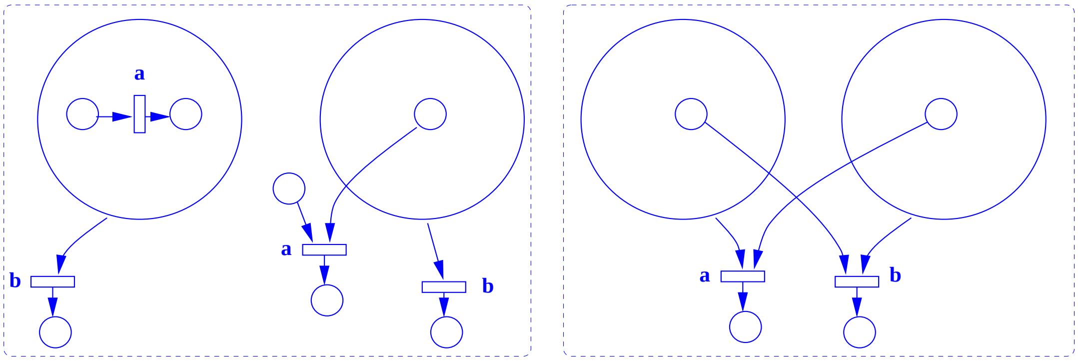


Figure 12: Dominance and cross-dominance of events

As Figure 12 shows it is possible that two events dominate each other. If b dominates a, but a does not dominate b, then we consider b as a transition with a higher vertical priority. If b dominates a and vice versa, then a and b are transitions with the same level of priority.

Definition 6.2 (Vertical priority in PCNs) *Transition b has higher vertical priority than transition a (denoted, $a \lhd b$) if*

$$a \unlhd b \ \wedge \ \neg(b \unlhd a).$$

Otherwise, a and b have the same vertical priority.

Hence, a and b have the same vertical priority iff $(a \unlhd b) \equiv (b \unlhd a)$.

Given a PCN one can uniquely define relation $\lhd$. However, to distinguish a PCN with vertical priorities (i.e., a PCNs whose behavior is interpreted under asynchronous semantics with priorities) from the same PCN without priorities (asynchronous semantics) we denote the former – $(N, \lhd)$, and the latter – N.

Similar to a case of PNs we define behavior of a PCN with vertical priorities through a structurally the same PCN without priorities.

Definition 6.3 (Enabling in PCNs with vertical priorities) *Let $(N, \lhd)$ be a PCN with vertical priorities, then transition a is enabled at m iff*

(1) a is enabled at m for N and

(2) there is no b enabled at m for N such that $a \lhd b$.

Now we are ready to define synthesis problems:

Problem 3 (Hierarchical synthesis with priorities for preemption): Given a TS (i.e., a TS without priority relation) build a PCN with vertical priorities $(N, \lhd)$ bisimilar to the TS.

Problem 4 (Hierarchical synthesis for a TS with given priorities) Given a $(TS, \lhd)$ (i.e., a TS with a priority relation) build a PCN, $(N, \lhd)$, with vertical priorities such that:

(1) $(N, \lhd)$ is bisimilar to the TS and
(2a) for any pair of events, a, b: $a \lhd b$ iff $a \triangleleft b$
or another weaker and probably more practical version:
(2b) $a \lhd b \Rightarrow a \triangleleft b$.

Informally, a preemption is allowed only for the transitions which has higher than internal transitions ("flat") priorities in the $(TS, \triangleleft)$.

6.4 Vertical priorities for TSs

To describe solutions for problems 3 and 4 we first need to define precisely vertical priorities in terms of TSs. We can do it by projecting the corresponding definitions in terms of region charts. To denote the corresponding relations we will use the same notation as for transitions in PCNs with vertical priorities.

Definition 6.4 (Dominance relation in TSs) *Let $TS = (S, E, T, s_{in})$ be a* JECTS *and R is a set of region charts derived for the TS. Then event b* dominates *event a (denoted, $a \unlhd b$) if*

$$\exists rc \in R : rc \in {}^{\bullet}a \;\wedge\; rc \in {}^{\circ}b.$$

In other words, region chart rc, which is a pre-region for a, can be preempted or self-preempted by b.

Definition 6.5 (Vertical priority in TSs) *Let $TS = (S, E, T, s_{in})$ be a* JECTS *and R is a set of region charts derived for the TS. Event b has higher vertical priority than event a (denoted, $a \lhd b$) if*

$$a \unlhd b \;\wedge\; \neg(b \unlhd a).$$

Otherwise, a and b have the same vertical priority.

6.5 Synthesis of PCNs with vertical priorities. Solution

For solving problem 3 the excitation closure condition should be modified as follows:

$$GER(a) = \bigcap_{rc \in {}^{\bullet}a} rc - \bigcup_{a \lhd b} GER(b),$$

i.e. from the intersection of preregions of a we need to exclude excitation regions of those events which have higher vertical priority than b. An example of synthesis of a PCN with vertical priorities is shown in Figure 13. Note that $TS1$ and $TS2$ produces exactly the same PCN. However, for $TS1$ it must be interpreted without priorities, while for $TS2$ – with vertical priorities.

Problem 4 is a modification of Problem 3 under constraints on generation of preemptive transitions.

6.6 Power of PCNs with priorities

In this section we show that PCNs with vertical priorities are Turing complete.

Theorem 6.1 PCN*s with* vertical priorities *are Turing complete.*

Proof: In [1, 32] it is shown that inhibitor nets can model Shepardson's register machines (which are Turing complete) and therefore inhibitor nets are Turing complete. A register machine has a finite number of registers of unlimited capacity. It has only three instructions: $Inc(Rn)$ – increment register Rn by one, $Dec(Rn)$ – if $Rn > 0$, then decrement register Rn by one, and $JE(Rn, s)$ – if $Rn = 0$, then jump to instruction s. Instructions Inc and Dec are trivially modeled by ordinary PNs. Instruction JE is easily modeled with inhibitor nets. Figure 14 shows how JE instruction can be modeled with an inhibitor net and with a PCN with vertical priorities. Priorities are essential to guarantee that jump, JE, does not occur

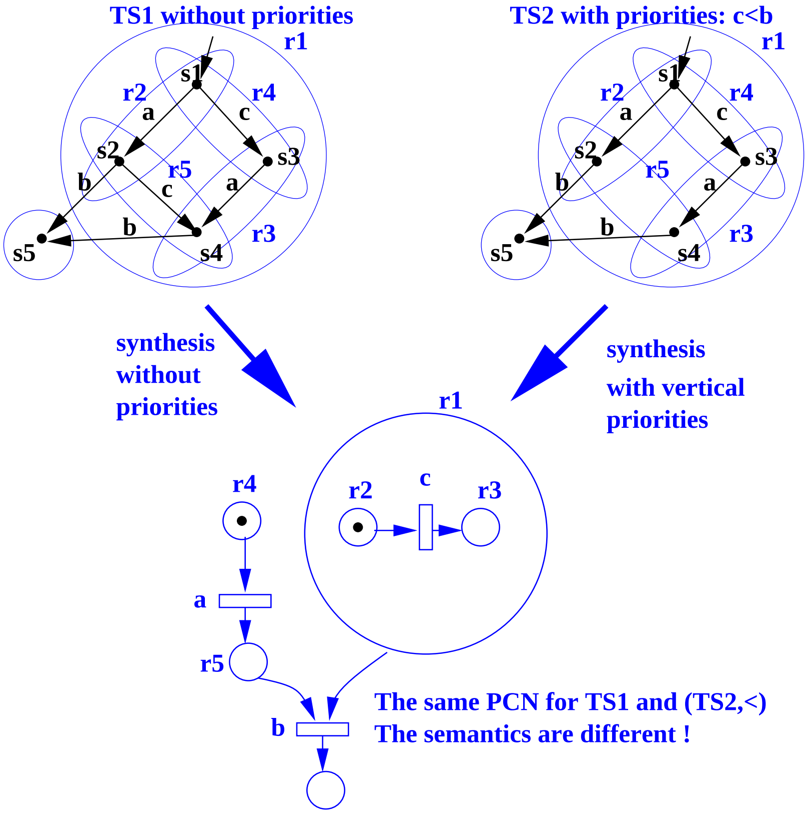


Figure 13: Synthesis with and without vertical priorities. An example.

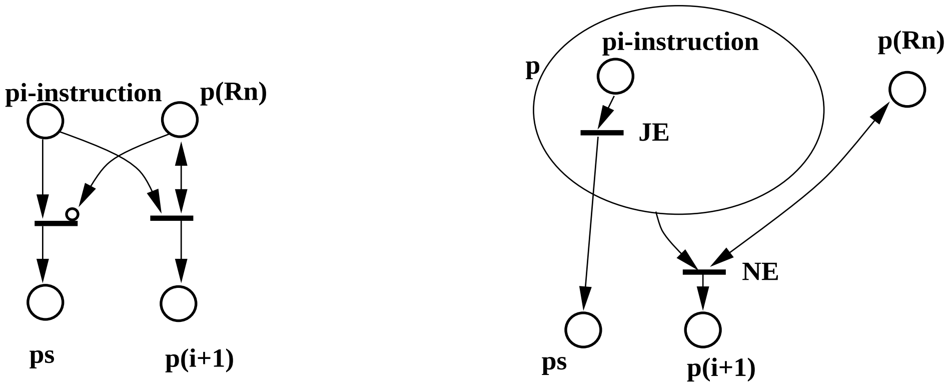


Figure 14: Modeling register machine's instruction JE with inhibitor nets and PCNs.

if p_{Rn} contains more than zero token. If $Rn = 0$, then place p_{Rn} contains no tokens. The only enabled transition is JE. Its firing leads to a token in place ps – triggering instruction s. If $Rn > 0$, then place p_{Rn} contains some tokens and a higher priority preemptive transition NE will fire triggering instruction $i+1$. Note that an arc $p \rightarrow NE$ serves only for establishing a higher priority of NE in case $Rn > 0$. $\square$

This theorem shows that vertical priorities has the same modeling power as horizontal priorities.

7 Conclusion

We have introduced a model, Place Chart Nets (PCNs), which combines the features of Petri nets and State Charts. Unlike Petri nets, PCNs allow hierarchy and succinct modelling of interrupt and abortion of activity in the systems. Contrary to State Charts or other hierarchical models, the notion of hierarchy in PCNs is focused on the asynchronous model of preemption. The main application for such a model is deemed to be the design of complex reactive systems, whose behavior could generally be asynchronous. We have formally defined preemption at the structural level in PCNs, and showed that modeling a similar behavior in PNs would imply an exponential blow-up in the size of the model. Although bounded PCNs (finite case) can be modeled by behaviorally equivalent PNs, our analysis of the infinite case has shown that the class of PCN languages properly includes that of PN languages. We have presented a method for the synthesis of safe PCNs from (flat) transition systems, which is based on the use of ideas from the theory of regions. Finally, we have been able to extend those results for synthesis of bounded PCNs and systems with priorities. We envisage potential directions for further research in the development of PCN-based methods for synthesis of asynchronous or mixed synchronous-asynchronous systems and embedded reactive systems, "co-implemented" in software and hardware.

Acknowledgments. We thank Juan M. Pérez for implementing the tool for traversal of PCNs.

References

[1] T. Agerwala. A complete model for representing the coordination of asynchronous processes. Hopkins Computer Research Report 32, Johns Hopkins University, July 1974.

[2] T. Araki and T. Kasami. Some decision problems related to the reachability problem for petri nets. *Theor. Computer Science*, 3:85–104, 1977.

[3] A. Arnold. *Finite Transition Systems.* Prentice Hall, 1994.

[4] A. Arnold and M. La Heux. Vector addition systems and semi-dyck languages. Technical Report Publications 78, Universite Lille, France, 1976.

[5] E. Badouel, L. Bernardinello, and Ph. Darondeau. Polynomial algorithms for the synthesis of bounded nets. Technical Report 2316, INRIA, RENNES Cedex, France, 1994.

[6] L. Bernardinello and F. De Cindio. A survey of basic net models and modular net classes. In *Lecture Notes in Computer Science*, volume 609, pages 304–351, 1992.

[7] L. Bernardinello, G. De Michelis, K. Petruni, and S. Vigna. On synchronic structure of transition systems. Technical report, Universita di Milano, Milano, 1994.

[8] G. Berry, P. Couronné, and G. Gonthier. The synchronous approach to reactive and real-time systems. *IEEE Proceedings*, 79, September 1991.

[9] G. Berry and H. Touati. Optimized controller synthesis using ESTEREL. In *Proceedings of the International Workshop on Logic Synthesis*, May 1993.

[10] Sh. S. Bhattacharyya, P. K. Murthy, and E. A. Lee. *Software Synthesis from Dataflow Graphs.* Kluwer Academic Publishers, 1996.

[11] J. Buck, S. Ha, E.A. Lee, and D.G. Messerschmitt. Ptolemy: a framework for simulating and prototyping heterogeneous systems. *Interntional Journal of Computer Simulation*, special issue on Simulation Software Development, January 1990.

[12] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Synthesizing Petri nets from state-based models. In *Proceedings of the International Conference on Computer-Aided Design*, pages 164–171, November 1995.

[13] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. Technical Report UPC-DAC-1996-19, Dept. of Computer Architecture, Universitat Politècnica de Catalunya, June 1996.

[14] J. Desel and W. Reisig. The synthesis problem of Petri nets. Technical Report TUM-I9231, Technische Universität München, September 1992.

[15] D. Drusinski and D. Harel. On the power of bounded concurrency. I. Finite automata. *Journal of the Association for Computing Machinery*, 41(3):517–539, May 1994.

[16] D. Drusinsky. Visually designing embedded-systems applications. *Dr.Dobb's Journal*, June 1995.

[17] Best E. and Koutny M. Petri net semantics of priority systems. *Theoretical Computer Science*, 96:175–215, 1992.

[18] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures. Part I, II. *Acta Informatica*, 27:315–368, 1990.

[19] A. Giua. Petri Net techniques for supervisory control of discrete event systems. In *Proc. of the First International Workshop on Manufacturing and Petri Nets*, pages 1–30, Osaka, Japan, June 25, 1996.

[20] M. H. T. Hack. Petri net languages. Technical Report TR-159, MIT, Laboratory of Computer Science, 1976.

[21] N. Halbwachs. *Synchronous Programming of reactive systems.* Kluwer Academic Publishers, 1996.

[22] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[23] B. Heinemann. Subclasses of self-modifying nets. In *European Workshop on Application and Theory of Petri Nets 1980. Selected Papers from the First and Second European Workshop on Application and Theory of Petri Nets*, pages 187–192. Springer-Verlag, 1982.

[24] T. Holvoet and P. Verbaeten. Petri Charts: an alternative technique for hierarchical net construction. In *IEEE Conference on Sytems, Man and Cybernetics*, 1995.

[25] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[26] M. Jantzen. Language theory of Petri nets. In *Advances in Petri Nets 1986, Lecture Notes in Computer Science, Vol. 254-I*, pages 397–412. Springer-Verlag, 1987.

[27] G. Kahn. The semantics of a simple language for parallel programming. In *Proceedings of IFIP Congress*, August 1974.

[28] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3:147–195, 1969.

[29] M. Kishinevsky, J. Cortadella, A. Kondratyev, L. Lavagno, and A. Yakovlev. Synthesis of General Petri Nets. Technical Report 57, IEICE, Japan, May 1996.

[30] M. Kishinevsky, A. Kondratyev, A. Taubin, and V. Varshavsky. *Concurrent Hardware: The Theory and Practice of Self-Timed Design.* John Wiley and Sons, London, 1993.

[31] M. Nielsen, G. Rozenberg, and P.S. Thiagarajan. Elementary transition systems. *Theoretical Computer Science*, 96:3–33, 1992.

[32] J. L. Peterson. *Petri Net Theory and the modeling of systems.* Prentice-Hall, Inc., 1981.

[33] R. Valk. Self-modifying nets, a natural extension of petri nets. In *Lecture Notes in Computer Science. Automata, Languages and Programming*, volume 62, pages 464–476, 1978.

[34] R. Valk. Generalizations of petri nets. In *Lecture Notes in Computer Science. Mathematical Foundation of Computer Science, Proc. of the 10th Symp*, volume 118, pages 140–155, 1981.

[35] R. Valk. On processes of object Petri nets. Technical Report 185/96, University of Hamburg, Computer Science Department, 1996.

8 Appendix I: Proofs of the main statements

Theorem 4.1

Proof: Based on the theory of regions to synthesize safe PNs from finite transition systems [12, 13], we need to find a set of regions such that the intersection of the pre-regions of each event e is exactly $\mathsf{GER}(e)$ (see section 5). In case this property cannot be fulfilled, label splitting will have to be applied, i.e. different arcs with the same label e will have to be renamed into $e_1, \ldots, e_k$ to allow excitation closure to hold. Without loss of generality we will prove that all arcs with label q in Figure 5.b have to be renamed into $q_1, \ldots, q_4$ to derive a safe PN for $n = 2$.

Let us take the transition system of Figure 5.b. To derive any pre-region r for event t_1 it is required that $s_1, s_3 \in r$ and $s_2, s_4 \notin r$. However event q violates the conditions of a region, since it is entering and not crossing r at the same time. The only way to satisfy the excitation closure for t_1 is to split the labels of the arcs with label q as follows: $s_1 \xrightarrow{q_1} s_1$, $s_3 \xrightarrow{q_1} s_1$, $s_2 \xrightarrow{q_2} s_1$ and $s_4 \xrightarrow{q_2} s_1$. Similarly, to satisfy the excitation closure for t_2 it is required to split the labels q_1 and q_2 in the other dimension: $s_1 \xrightarrow{q_1} s_1$, $s_3 \xrightarrow{q_3} s_1$, $s_2 \xrightarrow{q_2} s_1$ and $s_4 \xrightarrow{q_4} s_1$. This derives a non-pure PN with four different transitions with label q.

The reader can easily prove that splitting labels t_1 and t_2 in the TS does not help to avoid a complete splitting of q. The proof can also be trivially generalized for n concurrent transitions, thus requiring 2^n different transitions with label q. □

Theorem 4.2

For the proof of this theorem and the following we will use two pumping lemmas for RL and CFL.
Reminder:

- **Pumping lemma for RL**: Let L be a regular language. Then there is a constant n such that if $z \in L$ and $|z| \geq n$, then we may write $z = uvw$ in such a way that:

$$|uv| \leq n \wedge |v| \geq 1 \wedge \forall i \geq 0 : uv^i w \in L$$

- **Pumping lemma for CFL**: Let L be a context-free language. Then there is a constant n such that if $z \in L$ and $|z| \geq n$, then we may write $z = uvwxy$ in such a way that:

$$|vwx| \leq n \wedge |vx| \geq 1 \wedge \forall i \geq 0 : uv^i wx^i y \in L$$

Proof: Let $L = \{a^i b^j | i \geq j \geq 0\}$. Clearly, $L \in \mathcal{P} \subset \mathcal{P}^\lambda$ since it can be generated by the PN N in Figure 6.a. Assume that $L* = \{(a^i b^j) * | i \geq j \geq 0\} \in \mathcal{P}^\lambda$.

(1) Let us first show that $L* \notin RL$. Assume $L*$ is regular and let n be the integer in the pumping lemma for RL. Let $z = a^n b^n$. By the pumping lemma z may be written as uvw, where $|uv| \leq n$ and $|v| \geq 1$. Hence, v contains only a's and it contain at least one a. For all $i \geq 0$ $uv^i w \in L*$. In particular, let $i = 0$. Then $uw \in L*$. However, uw contains less a's then b's and therefore is not in $L*$, a contradiction. We conclude that $L*$ is not regular.

(2) By assumption $L* \in \mathcal{P}^\lambda$. Since, $\mathcal{P}^\lambda \subset \mathcal{L}^\lambda$, it follows that $L* \in \mathcal{L}^\lambda$. Let $\mathcal{L}*^\lambda$ be a subclass of languages from $\mathcal{L}^\lambda$ closed under the Kleene closure. Since $L*$ is obviously closed under the Kleene closure it belongs to $\mathcal{L}*^\lambda$.

(3) From **(2)** and the fact that L-type PNL closed under the Klene closure are in RL [32] it follows that $L* \in RL$. However, in **(1)** we proved that $L* \notin RL$, a contradiction. Hence, $L* \notin \mathcal{P}^\lambda$. Moreover, from the proof it follows that $L* \notin \mathcal{L}^\lambda$.

(4) The zero marking is never reached in N. Therefore applying construction from Lemma 4.1 it is trivial to show that $L* \in \mathcal{PC}^\lambda$.

Hence, we proved items (1) and (3) of the theorem. Item (2) follows from the inclusion $\mathcal{PC}^\lambda \subset \mathcal{LC}^\lambda$. □

Lemma 4.2

Proof: The proof will be given by showing an example of such a language. Let $L = \{a^i b^j c^k | i \geq j \geq k \geq 0\}$. Easy to show that $L \in \mathcal{P}$. For our purposes it is even enough to show that $L \in \mathcal{P}^\lambda$. The PN generating this P-type language is shown in Figure 15.

By applying construction from Lemma 4.1 one can build a PCN generating the Kleene closure of L: $L* = \{(a^i b^j c^k) * | i \geq j \geq k \geq 0\}$. Obviously, $L*$ does not belong to $\mathcal{P}^\lambda$ and therefore $L* \in \mathcal{PC}^\lambda - \mathcal{P}^\lambda$.

Now let us prove that $L* \notin CFL$. This is a simple exercise in application of the pumping lemma for CFL.

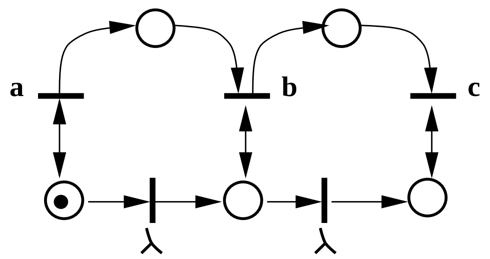


Figure 15: PN generating P-type language $L = \{a^i b^j c^k | i \geq j \geq k \geq 0\}$

Suppose $L*$ is context-free and let n be the constant of the pumping lemma for CFL. Consider $z = a^n b^n c^n$. Write z as $uvwxy$. Try to find a place where the pumped substrings v and x could lie in $a^n b^n c^n$.

Since $|vx| \geq 1$, there are seven cases for vx to contain instances of symbols: (1) a's, b's, and c's; (2) a's, and c's; (3) a's, and b's; (4) b's, and c's; (5) only a's; (6) only b's; (7) only c's.

Since $|vwx| \leq n$, it is not possible for vx to contain instances of a's and c's, because the rightmost a is $n+1$ positions away from the leftmost c. Hence, cases (1) and (2) are not possible.

If vx consists of a's only, then uwy (the string $uv^i wx^i y$ with $i = 0$) has less a's than b's and c's (since $|vx| \geq 1$) and hence $uwy \notin L*$, a contradiction. Hence, case (5) is not possible.

Similarly, if vx consists of b's only, then uwy has less b's than c's; and if vx consists of a's and b's, then uwy has less a's and b's than c's. Hence, cases (6) and (3) are not possible as well.

If vx consists of c's only, then let us consider a string $uv^n wx^n y$ which should be in $L*$ by condition of the pumping lemma. However, it has more c's than a's and b's and therefore is not in $L*$, a contradiction. Case (7) is not possible.

Similarly, the last case (4) is not possible, since $uv^n wx^n y$ contains more c's and b's than a's. We conclude that $L*$ is not a CF language. □

Lemma 5.2

Proof: Case 1. Let $e \in rc_1^\circ$. Then $\texttt{EXIT}(e, rc_2) \vee \texttt{IN}(e, rc_2)$. Let $\texttt{EXIT}(e, rc_2)$ holds. For any transition of event e, (s, e, s'), state $s \in rc_3$. Let (s, e, s') and (s_1, e, s_1') are two arbitrary transitions of e. Both s' and s_1' are either inside rc_3 or outside rc_3. Really, if $s' \in rc_3$ and $s_1' \notin rc_3$, then $\texttt{in}(e, rc_3) \wedge \texttt{exit}(e, rc_3)$ and rc_3 is not a region chart. Hence, condition $\texttt{EXIT}(e, rc_3) \vee \texttt{IN}(e, rc_3)$ holds.

If $\texttt{IN}(e, rc_2)$, then also $\texttt{IN}(e, rc_3)$.

Case 2. Let $e \in {}^\circ rc_1^\circ$. Then by Lemma 5.1 predicate $\texttt{IN}(e, rc_2)$ holds. Since $rc_2 \subset rc_3$, also $\texttt{IN}(e, rc_3)$ holds. In all the cases $\texttt{EXIT}(e, rc_3) \vee \texttt{IN}(e, rc_3)$ holds and e is neither preemptive nor self-preemptive for rc_3. □

Lemma 5.3

Proof: Since s is not terminal, there is an event e such that $s \in GER(e)$. By the excitation closure, there is a region chart $rc_1 \in R$ such that $s \in rc_1$. If $bottom(rc_1)$ we are done. Otherwise, by the inner closure, $\exists rc_2 \in R : rc_2 \prec\!\prec rc_1 \wedge s \in rc_2$ and then the recursion is applied. This recursion is finite since at each step rc_i contains less states than rc_{i-1} and hence at some step the bottom region chart rc_k will be reached. □

Lemma 5.4

Proof: From Lemma 5.3 it follows that every state s is covered by some bottom region chart $rb \in R$. Since the justification condition is satisfied – every bottom region chart is covered by some top region chart. Hence, every state is covered by a top chart. □

Lemma 5.5

Proof: Let $e \in rc_1^\circ \ \wedge \ e \notin rc_3^{\bigcirc}$, then by Definition 5.3 (item 2) it follows that $\mathtt{EXIT}(e, rc_3) \ \vee \ \mathtt{IN}(e, rc_3)$. If $\mathtt{EXIT}(e, rc_3)$, then e is preemptive for rc_2, since the preemption condition holds:

$$\mathtt{exit}(e, rc_2) \ \wedge \ \mathtt{out}(e, rc_2) \ \wedge \ \neg\mathtt{enter}(e, rc_2) \ \wedge \ \neg\mathtt{in}(e, rc_2).$$

If $\mathtt{IN}(e, rc_3)$ is true, then e is self-preemptive for rc_2, since the self-preemptive condition holds:

$$\mathtt{enter}(e, rc_2) \ \wedge \ \mathtt{in}(e, rc_2) \ \wedge \ \neg\mathtt{out}(e, rc_2) \ \wedge \ \neg\mathtt{exit}(e, rc_2).$$

In both cases $e \in rc_2^{\bigcirc}$.

Assume now that $e \in {}^\circ rc_1^\circ \ \wedge \ e \notin rc_3^{\bigcirc}$. Due to Lemma 5.1 predicate $\mathtt{IN}(e, rc_3)$ holds. Therefore,

$$\mathtt{exit}(e, rc_2) \ \wedge \ \mathtt{out}(e, rc_2) \ \wedge \ \neg\mathtt{enter}(e, rc_2) \ \wedge \ \neg\mathtt{in}(e, rc_2).$$

and e is preemptive for rc_2. □

Lemma 5.6

Proof: The first condition of justification, $rc_2 \subset rc$, follows immediately from the definition of rc_2. Hence we need to prove that (1) $rc_2 = rc - rc_1$ is a region chart, (2) the second and the third condition of justification of rc_2 by rc are satisfied (see Definition 5.3).

For an arbitrary event e five cases are possible with respect to region chart rc_1:

1. e exits rc_1,
2. e enters rc_1,
3. e does not cross rc_1,
4. e preempts rc_1,
5. e self-preempts rc_1.

Let us resume by cases (illustrated with Figure 16).

1. $\mathtt{EXIT}(e, rc_1) \Rightarrow \mathtt{EXIT}(e, rc) \ \vee \ \mathtt{IN}(e, rc)$. The only alternative $\mathtt{exit}(e, rc) \ \wedge \ \mathtt{in}(e, rc)$ is not possible since this would violate region chart conditions from Definition 5.2 for the chart rc. If $\mathtt{EXIT}(e, rc)$, then $\mathtt{OUT}(e, rc_2)$. If $\mathtt{IN}(e, rc)$, then $\mathtt{ENTER}(e, rc_2)$. In both cases region condition (i) from Definition 5.2 for rc_2 is not violated by e (cf. Figure 16(1).)

2. $\mathtt{ENTER}(e, rc_1) \Rightarrow \mathtt{ENTER}(e, rc) \ \vee \ \mathtt{IN}(e, rc) \ \vee \ e \in {}^\circ rc^\circ$. Three possible cases are presented in Figure 16(2). In the first case $\mathtt{ENTER}(e, rc)$ and hence $\mathtt{OUT}(e, rc_2)$ holds. In the second – $\mathtt{IN}(e, rc)$ and hence $\mathtt{EXIT}(e, rc_2)$ is satisfied. In the third case e is self-preemptive for rc and is preemptive for rc_2 [16]. In all three cases there is no violations of region chart condition for rc_2 and justification condition $rc_2 \prec rc$.

3. If e does not cross rc_1, then the following cases possible.

If $\mathtt{IN}(e, rc_1)$, then $\mathtt{IN}(e, rc) \ \wedge \ \mathtt{OUT}(e, rc_2)$ – no region violations for rc_2.

If $\mathtt{OUT}(e, rc_1)$, then three cases are possible (cf. Figure 16(3),a,b,c). In the first case e does not cross rc and rc_2. In the second, e is self-preemptive both for rc and rc_2. In the third one, e is preemptive both for rc and rc_2. The latter two cases does not violate neither region chart condition for rc_2 nor justification conditions $rc_2 \prec rc$ and $rc_1 \prec rc$, since e is outside rc_1.

[16] Note: this is the case when a new self-preemption may appear in the parent rc. Item (3) of Definition5.3 is applied.

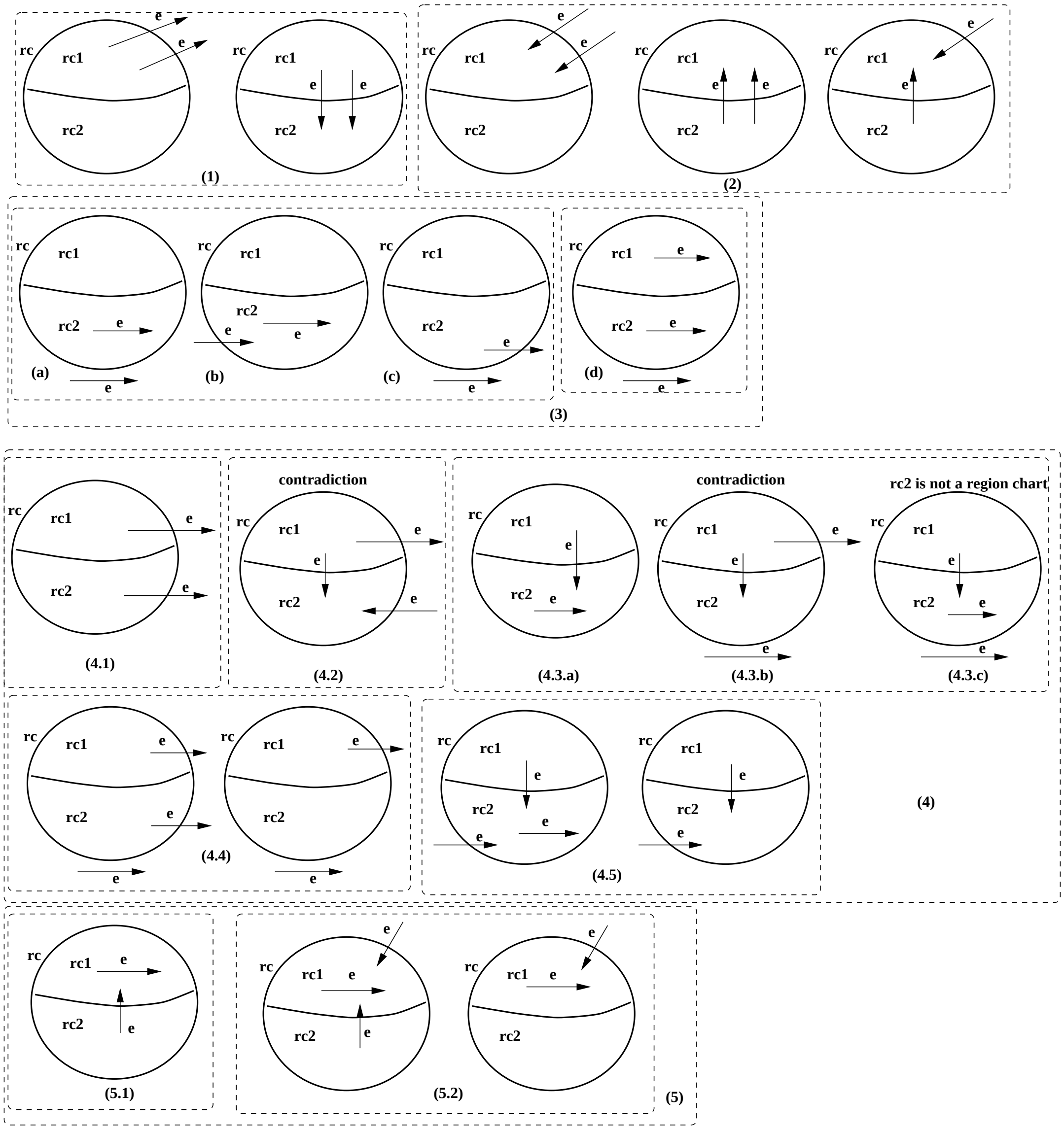


Figure 16: Different cases for a descendant complement

If $\texttt{in}(e, rc_1) \ \wedge \ \texttt{out}(e, rc_1)$, then the only case possible is represented by Figure 16(3),d. There are no violations of region condition (i) by event e for rc and rc_2. No transition (s, e, s') can cross the border of rc since this would make e self-preemptive or preemptive for rc. Since neither $\texttt{OUT}(e, rc_1)$, no $\texttt{ENTER}(e, rc_1)$ it contradicts the third condition of justification for rc_1.

4. Event e is preemptive for rc_1. Five sub-cases are possible depending on the relation of e and region chart rc.

1. Let $\texttt{EXIT}(e, rc)$ is true. Then e is preemptive for rc_2 (cf. Figure 16(4.1)).
2. Predicate $\texttt{ENTER}(e, rc)$ cannot be true, since it would contradict $\texttt{exit}(e, rc_1)$. Hence this case is not possible (cf. Figure 16(4.2)).
3. Let e does not cross rc. Consider the following three sub-cases: (a) $\texttt{IN}(e, rc)$, (b) $\texttt{OUT}(e, rc)$, and (c) $\texttt{in}(e, rc) \ \wedge \ \texttt{out}(e, rc) \ \wedge \ \neg\texttt{enter}(e, rc) \ \wedge \ \neg\texttt{exit}(e, rc)$.

   Sub-case (a) (cf. Figure 16(4.3.a.)) implies that e is self-preemptive for rc_2. Condition (iii) of Definition 5.2 is satisfied. Sub-case (b) is not possible. It contradicts condition $\texttt{exit}(e, rc_1)$ which must hold, since e is preemptive for rc_1 (cf. Figure 16(4.3.b)). Sub-case (c) is not possible either since it contradicts the second condition of Definition 5.3 for justification of region charts [17]. Really, event e is preemptive for rc_1 and satisfies region conditions for rc, so it must satisfy $\mathtt{IN}(e, rc) \vee \mathtt{EXIT}(e, rc)$. Note that if this condition would be omitted in Definition 5.3, then as shown by Figure 16(4.3.c) set rc_2 would not necessarily be a region chart.

4. Let e be preemptive for rc. Then e is either also preemptive for rc_2 or is $\mathtt{OUT}(e, rc_2)$ (cf. Figure 16(4.4)). The second case satisfies the the third condition of justification (Definition 5.3).

5. Assume that e is self-preemptive for rc. Then $\neg\mathtt{exit}(e, rc)$ and only two cases depicted in Figure 16(4.5) are possible: (a) e is also self-preemptive for rc_2 and (b) $\mathtt{ENTER}(e, rc_2)$. The second case corresponds to the third condition of Definition 5.3.

5. Event e is self-preemptive for rc_1. Since condition $\mathtt{in}(e, rc_1)$ holds, also $\mathtt{in}(e, rc)$ is true. Therefore, predicates $\mathtt{EXIT}(e, rc)$ and $\mathtt{ENTER}(e, rc)$ are false and e cannot be preemptive for rc. So, we need to consider only two sub-cases depending on the relation of e and region chart rc.

1. Let e does not cross rc. Consider the following three sub-cases: (a) $\mathtt{IN}(e, rc)$ is satisfied. Then (cf. Figure 16(5.1)) e is preemptive for rc_2, (b) Predicate $\mathtt{OUT}(e, rc)$ cannot be satisfied, since $\mathtt{in}(e, rc)$ is true, and
(c)Condition $\mathtt{in}(e, rc) \wedge \mathtt{out}(e, rc) \wedge \neg\mathtt{enter}(e, rc) \wedge \neg\mathtt{exit}(e, rc)$ cannot be satisfied either, since condition e is self-preemptive for rc_1 implies $\neg\mathtt{out}(e, rc)$.

2. Let e is self-preemptive for rc. Then e is either preemptive for rc_2 or is $\mathtt{OUT}(e, rc_2)$ (cf. Figure 16(5.2)). If the former is true, then preemptive for rc_2 event e converts to a self-preemptive for rc. By Definition 5.3 this is legal for justification. If the latter is true, then the the third condition of justification (Definition 5.3) is satisfied.

We proved that rc_2 is a region chart. We also proved all conditions for justification of rc_2 by rc, except one: we did not show progress. To complete the proof of justification condition, one need to prove that there is at least one event e such that $e \in rc_2^{\bigcirc}$ and $e \notin rc^{\bigcirc}$. Due to Lemma 5.5 if preemption disappear in a parent rc, then it preempts both rc_1 and in rc_2. Formally, if $e \in rc_1^{\bigcirc}$ and $e \notin rc^{\bigcirc}$, then $e \in rc_2^{\bigcirc}$. Since rc_1 is justified by rc there must be at least one event e: $e \in rc_1^{\bigcirc} \ \wedge \ e \notin rc^{\bigcirc}$. For the same event e: $e \in rc_2^{\bigcirc} \ \wedge \ e \notin rc^{\bigcirc}$ [18] and hence $rc_2 \prec rc$. $\Box$

Lemma 5.7

Proof: The proof follows immediately from resuming all possible cases of the proof of Lemma 5.6. All

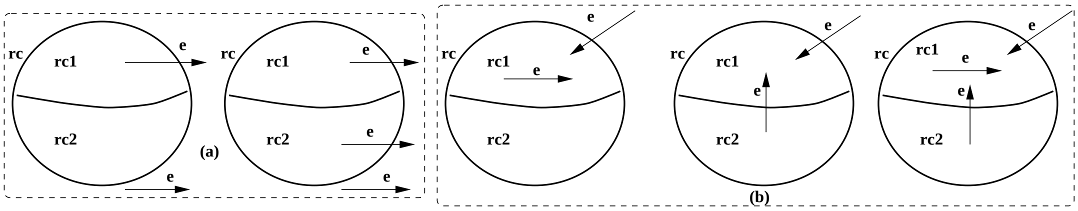


Figure 17: Preemption of non-bottom region chart

possible cases for $e \in rc^{\circ}$ are depicted in Figure 17.a and for $e \in {}^{\circ}rc^{\circ}$ – in Figure 17.b. $\Box$

[17] This is the case where we need to require that a justified event e must satisfy $IN(e, rc) \vee EXIT(e, rc)$.
[18] This can correspond only to cases 4.1, 5.1.a, or 4.3.a of the proof above.

Lemma 5.8

Proof: By definition of $\mathcal{R}$ we need to prove that $m = \rho(R_s)$ iff $m^b = \rho(R_s^b)$.

$\Rightarrow$. Let $m = \rho(R_s)$. Since there is a one-to-one correspondence between region charts and place charts for any $c \in m$: $bottom(c) \Leftrightarrow bottom(\rho^{-1}(c))$. Hence, $m^b = \rho(R_s^b)$.

$\Leftarrow$. Let $m^b = \rho(R_s^b)$. By definition of a marking: for any $c \in C$ if $c \in m$ and $c \notin m^b$, then there is at least one bottom place b: $b \in m^b$ and $b < c$. By construction of the net it follows that in the TS there is exactly one region chart corresponding to c, $\rho^{-1}(c)$, such that $\rho^{-1}(b) \prec \rho^{-1}(c)$. By definition of the justification relation: $\rho^{-1}(b) \subset \rho^{-1}(c)$. Inclusion $s \in \rho^{-1}(b)$ implies $s \in \rho^{-1}(c)$. Hence, we proved that if $c \in m$ then $\rho^{-1}(c) \in R_s$ and therefore $m \subseteq \rho(R_s)$.

Let us prove that the opposite also holds: $\rho(R_s) \subseteq m$. Assume the opposite, i.e., there is a region chart $rc \in R_s$ such that $\rho(rc) \notin m$. Obviously rc cannot be a bottom chart, otherwise the condition $m^b = \rho(R_s^b)$ would be violated. Since R satisfies the inner closure condition, Lemma 5.3 can be applied and every non-terminal state is covered by a bottom region chart. Hence, $\exists rb : rb \prec rc \ \wedge \ rb \in R_s$. Then place $\rho(rb) \in m$. By construction place chart $\rho(rc)$ belongs to net N_R and by definition of the marking: $\rho(rc) \in m$. We have reached a contradiction. $\Box$

Theorem 5.1

Proof: The bisimilarity between TS and $RG(N_R)$ will be proved with respect to relation $\mathcal{R}$ defined above.

1. First let us prove that $RG(N_R)$ simulates TS. We need to prove that given $s \xrightarrow{e} s_1$ and $s\mathcal{R}m$ for some marking m_1 two conditions hold: (1) $m \xrightarrow{e} m_1$ and (2) $s_1\mathcal{R}m_1$.

The proof will be done by induction on the length of the path between the s_0 and the current state s.

Basis. Let s_0 be the initial state of the TS and m_0 is the initial marking of the $RG(N_R)$. By construction (step 2 of the algorithm 1) $s_0\mathcal{R}m_0$.

Inductive hypothesis. Assume that for an arbitrary k if there is a path σ between the initial state s_0 and state s of a length $0 \leq |\sigma| \leq k$, then the statement is correct.

Inductive step. Let $s_0 \stackrel{\sigma}{\Rightarrow} s \wedge 0 \leq |\sigma| \leq k$, then by inductive hypothesis there is a marking m in $RG(N_R)$ such that $m_0 \stackrel{\sigma}{\Rightarrow} m$ and $s\mathcal{R}m$. Let $s \xrightarrow{e} s_1$.

(a). State s is not terminal since $s \in GER(e)$, hence (since excitation closure is satisfied) $\forall rc \in {}^\bullet e \cup {}^\star e^\star : s \in rc$. Furthermore, since $s\mathcal{R}m$ for any region chart $rc \in {}^\bullet e \cup {}^\star e^\star$ the corresponding place chart $\rho(rc) \in m$. By construction of the net N_R there is only one transition labeled with e and all input charts of e corresponds either to pre-regions of e or to preemptive loop pre-regions of e in the TS. Therefore, all input charts of e are marked in m and there a marking $m_1 : m \xrightarrow{e} m_1$.

(b). Let us prove that $m_1\mathcal{R}s_1$, i.e. either both s_1 and m_1 are terminal or $m_1 = \rho(R_{s_1})$. If s_1 is terminal, then $\forall e : s_1 \notin GER(e)$. Hence no transitions can be enabled in m_1 in the net N_R. Clearly m_1 is either an empty marking or a non-empty terminal marking. So, in this case: $s_1\mathcal{R}m_1$.

Let us assume that s_1 is not terminal. The TS is a JECTS and due to Lemma 5.8 it is enough to prove that $m_1^b = \rho(R_{s_1}^b)$. Set of bottom region charts R_s^b can be partitioned into the four disjoint subsets (some of them can be empty): $R_s = R_s^1 \cup R_s^2 \cup R_s^3 \cup R_s^4$, where $R_s^1 \subseteq {}^\bullet e$ is a set of pre-region bottom charts of e, $R_s^2 \subseteq e^\circ$ contains bottom charts preempted by e, R_s^3 contains bottom charts self-preempted by e and R_s^4 contains bottom charts which are not crossed by e.

For any $rb \in R_s^1 \cup R_s^2 : s_1 \notin rb$ while for any $rb \in R_s^3 \cup R_s^4 : s_1 \in rb$. Hence, $R_{s_1}^b = R_s^3 \cup R_s^4 \cup R_{s_1}^5 \cup R_{s_1}^6$, where $R_{s_1}^5$ contains post-region bottom charts for e and $R_{s_1}^6$ contains bottom charts self-preempted by e, which (contrary to R_s^3) do not cover s.

Let us examine marking m_1^b. By the firing rule all bottom places corresponding to pre-region bottom charts (set R_s^1) and bottom charts preempted by e (set R_s^2) are not included into m_1^b, while all bottom places from m^b that corresponds to bottom region charts from sets R_s^3 and R_s^4 are included into m_1^b, since firing of e cannot change the marking of those bottom places.

In addition m_1^b contains output bottom places of e, $c \in e^\bullet$. By construction of the flow relation of the net (step 3 of the algorithm) any such output place, c, is an image of the bottom region chart from the set of post-region bottom charts $R_{s_1}^5$ or from the set of bottom region charts self-preempted by e – $R_{s_1}^6$:

$$\exists rb \in R_{s_1}^5 \cup R_{s_1}^6 : c = \rho(rb).$$

Therefore, $m_1^b = \rho(R_{s_1}^b)$ and hence $m_1 \mathcal{R} s_1$.

2. Let us prove that TS simulates $RG(N_R)$ and N_R is safe and pure. Net N_R is pure by construction, since no self-loop arcs can appear if Algorithm 1 is applied.

Basis. By construction $m_0 \mathcal{R} s_0$ and m_0 is safe.

Inductive hypothesis. Assume that for an arbitrary k if there is a path σ between the initial marking m_0 and marking m of a length $0 \le |\sigma| \le k$, then the statement is correct.

Inductive step. Let in $RG(N_R)$: $m_0 \stackrel{\sigma}{\Rightarrow} m \ \wedge \ 0 \le |\sigma| \le k$, then by inductive hypothesis m is safe and there is a state s in the TS such that $s_0 \stackrel{\sigma}{\Rightarrow} s$, $m \mathcal{R} s$.

(a). Let $m \xrightarrow{e} m_1$. Then all input charts of e are in m. By construction of N_R the input place charts of e corresponds to the pre-region charts of e and the preemptive loop pre-regions of e in the TS. Note also that since R is excitation closed, ${}^\bullet e \cup {}^\star e^\star \neq \emptyset$ for any e and in the N_R each transition has at least one input place chart. Since $m \mathcal{R} s$ all region charts corresponding to place charts from m are included in R_s. Hence, all pre-region charts of e and all preemptive loop pre-regions of e are included in R_s.

The justified excitation closure condition is satisfied for the TS. Therefore, $s \in GER(e)$. Clearly, $\exists s_1 : s \xrightarrow{e} s_1$.

(b) Let us prove that marking m_1 is safe. Assume the opposite. Then there is a bottom place p such that $p \in m$, $p \in e\bullet$ and e delivers a second token into p. Since by the inductive hypothesis $m \mathcal{R} s$ state s satisfies the condition $s \in \rho^{-1}(p)$. On the other hand by construction $p \in e\bullet$ implies $\rho^{-1}(p) \in e^\bullet \ \vee \ \rho^{-1}(p) \in {}^\circ e^\circ$. Condition $(\mathtt{enter}(e, \rho^{-1}(p)) \ \wedge \ \mathtt{in}(e, \rho^{-1}(p)))$ is satisfied and therefore, event e must be preemptive for the bottom region chart $\rho^{-1}(p)$. Then by construction of the net N_R there is a place chart c such that $p < c \ \wedge \ c \in \bullet e$. When event e fires it empties all descendants of c, in particular p, and deliver one token to p. So, there is no way e can deliver a second token to the bottom place p and we have reached a contradiction with the initial assumption. We have proved that any reachable marking m_1 is safe and therefore the PCN N_R is safe.

(c) We need to prove now that $m_1 \mathcal{R} s_1$. If m_1 is an empty or a non-empty terminal marking, then s_1 is also terminal. Otherwise state s_1 would belong to some $GER(e)$ and condition $m_1 \xrightarrow{e}$ would hold.

Assume that m_1 is a non-terminal marking, then we need to prove that $m_1^b = \rho(R_{s_1}^b)$.

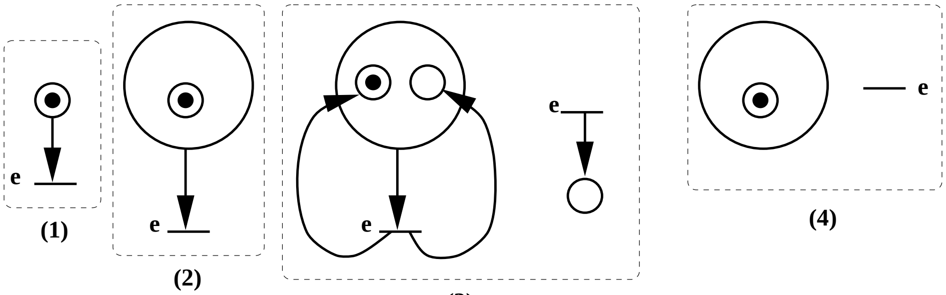


Figure 18: Bottom places in relation to e

Let us identify all bottom places whose marking is changed due to the decrementing and incrementing rules after transition e has fired. All bottom places of the net N_R which are marked in m or in m_1 can be partitioned into four disjoint subsets (some of them can be empty) as shown in Figure 18:

1. $m^1 = \{p : bottom(p) \ \wedge \ p \in \bullet e\}$ – input bottom places for e,
2. $m^2 = \{p : bottom(p) \ \wedge \ p \notin e \bullet \ \wedge \exists c : (p < c \ \wedge \ c \in \bullet e)\}$ – bottom places which are not output to e and that have an ancestor place chart input to e. These places are preempted by e.

3. $m^3 = \{p : \mathit{bottom}(p) \ \wedge \ p \in e\bullet\}$ – bottom places which are output to e. These places includes in particular self-preempted bottom places such that there is an ancestor, c: $p < c \ \wedge \ c \in \bullet e$.

4. $m^4 = \{p : p \in m \ \wedge \ \mathit{bottom}(p) \ \wedge \ p \notin e\bullet \ \wedge \forall c : p < c \Rightarrow c \notin \bullet e\}$ – bottom places that are marked in m and whose marking is not changed by firing e.

No input bottom places of e are marked in m_1, i.e., $\forall p \in m^1 : p \notin m_1^b$. In the TS: $\forall p \in m^1 : \rho^{-1}(p) \notin R_{s_1}$, since $\rho^{-1}(p)$ is a pre-region chart of e and condition $\neg\mathtt{in}(e, \rho^{-1}(p))$ must hold.

Similarly for any place from m^2: $p \notin m_1^b$ and for the TS: $\rho^{-1}(p) \notin R_{s_1}$, since $\rho^{-1}(p)$ covering s is preempted by e and condition $\neg\mathtt{in}(e, \rho^{-1}(p))$ is true. Hence it cannot cover s_1.

For any bottom place p from m^3: $p \in m_1^b$ and since N_R is safe, place p cannot contain more than one token in m_1. For the TS, by construction of the flow relation, the corresponding to p bottom region chart satisfies the following condition: $\rho^{-1}(p) \in R_{s_1}^b$.

For any bottom place, p, from m^4 marking cannot change after e has fired. The corresponding bottom region chart, $\rho^{-1}(p)$ covers both s and s_1: $\rho^{-1}(p) \in R_{s_1}^b$.

We can represent m_1^b and $R_{s_1}^b$ in the following form:

$$m_1^b = m^b - m^1 - m^2 \cup m^3$$

and

$$R_{s_1}^b = R_s^b - \rho^{-1}(m^1) - \rho^{-1}(m^2) \cup \rho^{-1}(m^3).$$

By inductive hypothesis $m^b = \rho(R_s^b)$ and hence $m_1^b = \rho(R_{s_1}^b)$. We conclude that $m_1 \mathcal{R} s_1$. $\Box$

Note that the proof above covers as a special case preemptive loops, $s \xrightarrow{e} s$, in TSs.

9 Appendix II: Flattening with silent transitions

In Section 4.1 we have shown that a PCN can be exponentially more succinct than bisimilar PN. We have shown it by applying a synthesis technique for PNs which is based on label splitting. Since the constructed PN is required to be bisimilar, no silent transitions are allowed in the PN. In this section we will show another (structural) technique for flattening that is based on inserting silent transitions for each marking in which a preemptive transition is enabled. This technique is called a sum-of-product flattening. Since silent transitions are allowed to be inserted, the resulting PN is weakly bisimilar to the original PN.

However both techniques (from Section 4.1 and from this one) give the same exponential complexity of the modeling PN and we conjecture that this bound is tight.

9.1 Flattening with atomic preemption

Figure 20 shows a PN which is weakly bisimilar to the PCN from Figure 19.

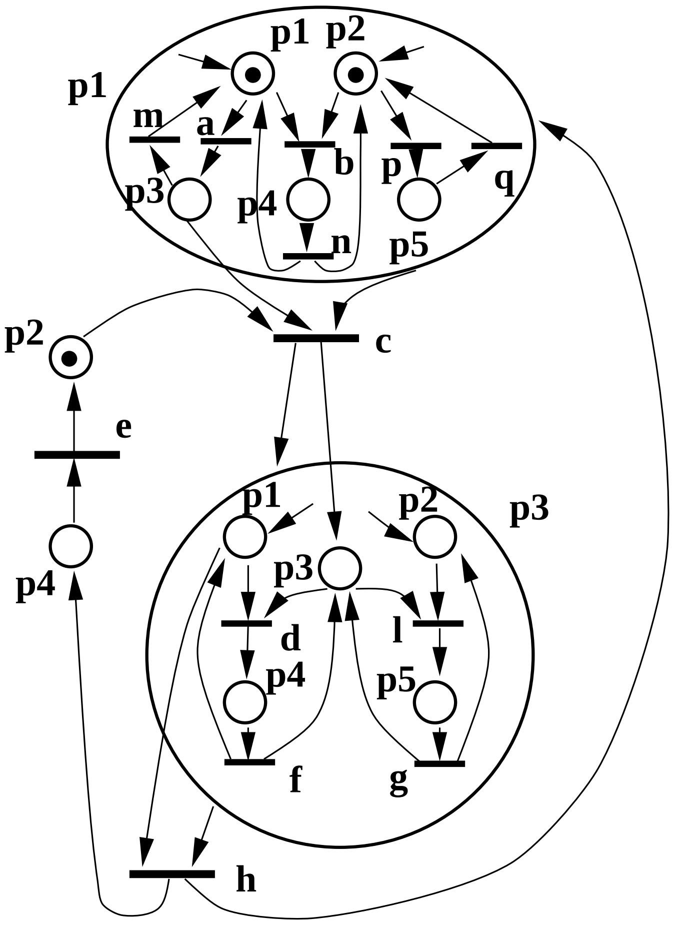


Figure 19: Example of PCN

There can be many different ways to build an equivalent PN. The method presented in Figure 20 is called a *sum-of-product flattening*, since it resembles a sum-of-product form for Boolean functions, where each additional silent transitions serves as a product and an additional place serves as a sum. The nice property of this method of flattening is that preemption is done in one step as an *atomic* action.

The process of SOP-flattening is recursive. Let k, $k > 0$, be a depth of hierarchy of the original PCN. Then at the first step a PCN with $(k-1)$ levels of hierarchy is build. It is done by removing all place charts at the level $(k-1)$ and simulating preemptive transitions exiting some place charts at the level $(k-1)$ with

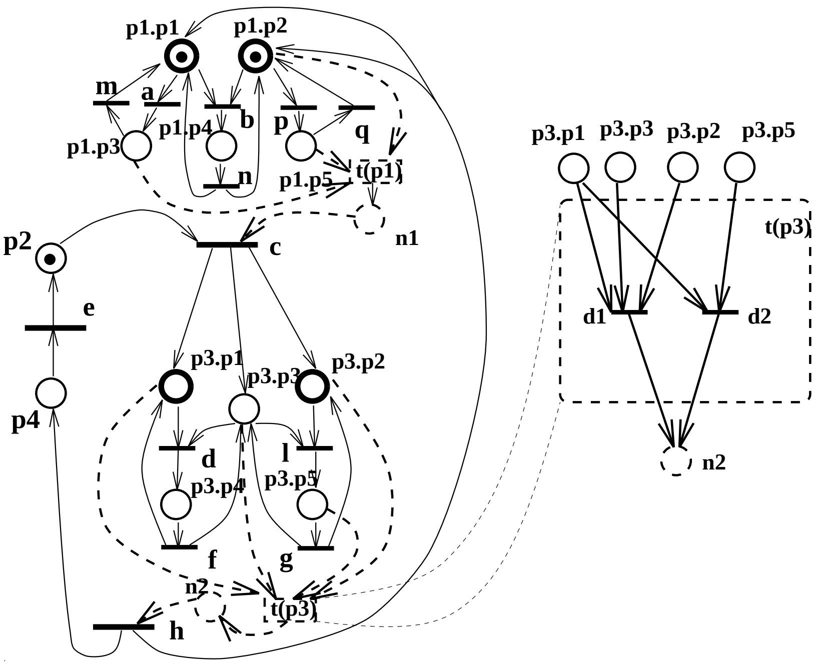


Figure 20: Flattening PCN with inserting silent transitions

a set of silent transitions. Each of these silent transitions corresponds to *one marking in the excitation region of the preemptive transition.* For example, for a place chart $p3$ two silent transitions, $d1$ and $d2$, are necessary with input places: $\{p3.p1, p3.p2, p3.p3\}$ and $\{p3.p1, p3.p5\}$, correspondingly, since only these two sets of places can be marked simultaneously with a $p3.p1$ – an ultimate enabling condition for preemptive transition h. As soon as one of the silent transitions $d1$, $d2$ has fired all places of the place chart $p3$ are immediately emptied.

At the next step the same procedure is applied to the obtained PCN with $k-1$ levels of hierarchy and so on ($k-1$ iterations).

Lemma 9.1 (Elimination of marking containment) *For any bounded* PCN *there is a weakly bisimilar* PCN*, such that for any two reachable markings, m_1 and m_2 the following condition holds: $m_1 \not\subset m_2$.*

Proof: (A sketch. Since a PCN is bounded by the condition of the theorem, for any two reachable markings, m_1 and m_2 such that $m_1 \stackrel{\sigma}{\Rightarrow} m_2$, the condition of the lemma holds $m_1 \not\subset m_2$. A transformation is required only for such pairs of markings that $m_1 \subset m_2$ and therefore $m_1 \stackrel{*}{\not\Rightarrow} m_2$.

Let us build a RG for the initial PCN and consider it as a TS. Let $[m_1]$ be a reachable closure for m_1, i.e. a set of markings which are reachable from m_1 in a RG ($[m_1] = \{m : m_1 \stackrel{\sigma_1}{\Rightarrow} m$). Clearly, $m_0 \notin [m_1]$ and $\forall m \in [m_1] : m \stackrel{*}{\not\Rightarrow} m_2$. Similarly $[m_2]$ is a reachable closure for m_2.

Assume first that $[m_1] \cap [m_2] = \emptyset$, then $[m_1]$ is a (redundant) region $r*$ or can be made a region by *new* event insertion (this might be needed if the border of $[m_1]$ intersects some entering events with violation of region conditions).

If $[m_1] \cap [m_2] \neq \emptyset$, then there are some paths from m_2 to m_1 or m_2, m_1 can be confluent. In such case again $[m_1]$ can cut some paths with violation of region conditions. Insert additional silent transitions to remove these region violations and to obtain a new region $r* : m_1 \in r* \wedge m_2 \notin r*$. After synthesis new region $r*$ will have a token in an extension of marking m_1, while it will have no token in the extension of m_2, therefore for m_1 and m_2 containment does not hold.

The idea of the method is illustrated with Figure 21. Initially, $m_1 = p3, m_2 = p2p3$. After transformation $m_1' = p^*p3, m_2' = p2p3$. $\Box$

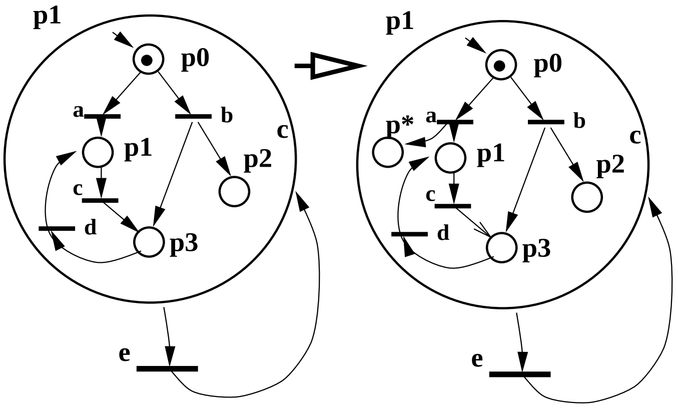


Figure 21: Eliminating marking containment

Theorem 9.1 (Reduction to PNs) *For any bounded* PCN *there is a weakly bisimilar* PN*, which can be constructed using sum-of-product flattening.*

Proof: (A sketch.) Since a PCN is bounded by Lemma 9.1 if an ER for a preemptive transition contains two markings m_1 and m_2 such that $m_1 \subset m_2$, then there is another bisimilar PCN without marking containments. We work with this transformed net. By the condition of the theorem, for any two reachable markings, m_1 and m_2 such that $m_1 \overset{\sigma}{\Rightarrow} m_2$, Since for any m_1,m_2: $m_1 \not\subset m_2$, any two silent transitions which model preemption are not mutually enabled at the same time and can never disable each other. Hence, when one of these silent transitions fires it decrements all non-empty places belonging to a place chart. This implies that two nets are weakly bisimilar. □
Since each PN is by definition a PCN, an opposite reduction is trivial.

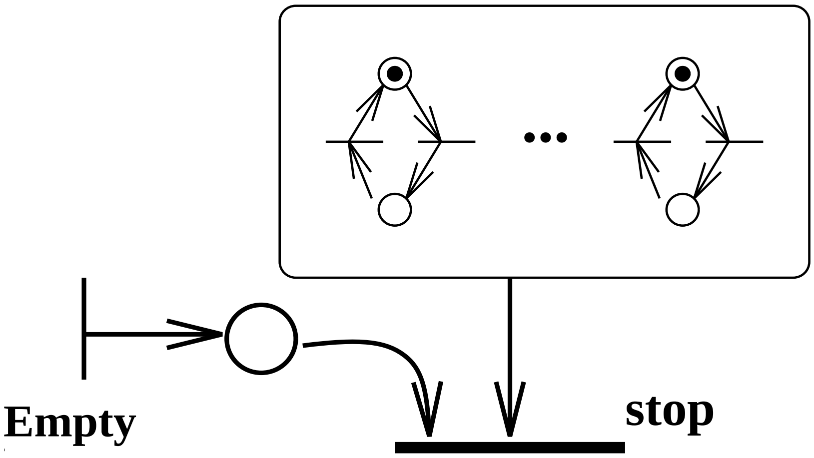


Figure 22: Exponential succinctness of PCNs

Let $||N||$ be a descriptive complexity of net N measured, for example, as a total number of places, transitions and arcs in the net N. In the worst case, for a subnet encapsulated in a safe place chart, the number of silent transitions required for SOP-flattening the place chart is growing linearly faster then an exponent from the number of children of this place chart as indicated by Figure 22. A PCN has complexity $O(m)$, while a SOP-flattered PN – $O(m * 2^m)$.

Theorem 9.2 (Complexity of the SOP-flattening) *Let N be a safe* PCN *and N' be a* PN *weakly bisimilar to N obtained by the SOP-flattening. Then the worst-case tight lower bound for complexity of N' is:*

$$||N'|| = \omega(||N|| * 2^{c*||N||})$$

Proof: This tight lower bound is valid due to the example in Figure 22. This PCN has m independent parallel processes with two simple places and two transitions each. They are encapsulated in a place chart with one output preemptive transition. The complexity of the PCN is $||N|| = O(m)||$. A SOP-flattered PN has 2^m silent transitions with m input places each, i.e., $||N'|| = O(m * 2^m) = O(||N|| * 2^{c*||N||}||$. $\Box$

That is, a PCN can be more than exponentially more succinct than a bisimilar PN obtained with the SOP-flattening.

9.2 Recursive example

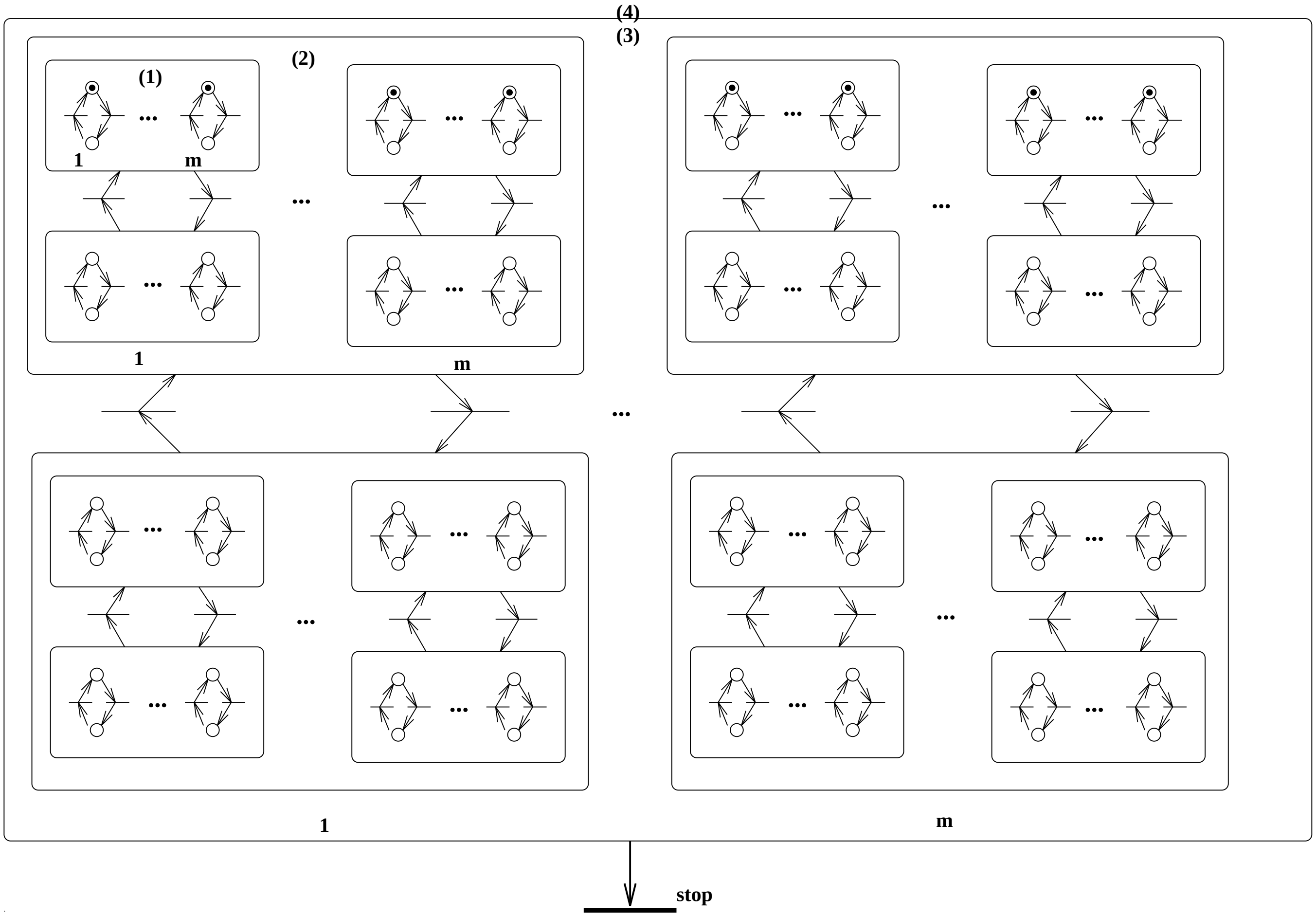


Figure 23: Recursive example

Let us consider as an exercise a recursive example in Figure 23. It illustrates a construction of the PCN, N_3, for a depth of hierarchy $k = 4$. At the top level there is only one place chart with one preemptive transition called *Stop*. At all previous levels the PCN follows a regular pattern. At the third level there are m independent parallel processes. Each of them contains two place charts and two preemptive transitions which toggle between these two place charts. Each of the place charts at the third level in its turn encapsulates m independent parallel processes with two place charts and two preemptive transitions toggling between these two place charts of the second level. Finally, each place chart of the second level encapsulates m bottom processes each containing two simple places and two transitions. At each level of hierarchy upper place charts are initial place charts for their parents.

Let us denote:

- N_k is an original PCN with a depth of hierarchy $k + 1$. Simple places are at the 0-level.

- N_{k-1} is obtained from N_k by flattening the the 1st (the lowest) level of the hierarchy.
- N_{k-2} is obtained from N_k by flattening the two lowest levels of hierarchy. It can be obtained from N_{k-1} by flattening its lowest level, and so on ...
- N_0 is a final flat net.

Let $Dum(N_{k-j})$ is the number of silent (dummy) transitions which are introduced into N_{k-j} due to the flattening of the jth level of hierarchy, and $DumArc(N_{k-j})$ is the number of input arcs to these silent (dummy) transitions.

The complexity of the original PCN, N_k is determined by the number of place charts, which in its turn is dominated by the number of simple places: $||N_k|| = P(N_k) = O(2m * (2m)^{k-1}) = O((2m)^k)$.

The number of markings in a excitation region of each preemptive transition at the 1st level of N_k is 2^m. Therefore, there must be 2^m silent transitions to simulate one preemptive transition, each of them with m input arcs. So, $Dum(N_{k-1}) = 2^m*(2m)^{k-1}$ and $DumArc(N_{k-1}) = O(m*2^m*(2m)^{k-1}) = O(2^m*2^{k-1}*m^k)$.

The complexity of N_{k-1} is:

$$||N_{k-1}|| = O(2^m * 2^{k-1} * m^k) + ||N_k||$$

In our example $||N_2|| = O(2^m * 2^2 * m^k) + (2m)^3$.

For the next iteration: The number of markings in a excitation region of each preemptive transition inside one place chart at the 1st level of N_{k_1} is 2^{m^2+m}. Therefore, there must be $2^{m^2+m} * (2m)^{k-2}$ silent transitions to simulate all preemptive transition at the 1st level of N_{k_1} , each of them with m^2 input arcs. So, $Dum(N_{k-2}) = O(m^2 * 2^{m^2+m} * (2m)^{k-2}) = O(2^{k-2} * m^k * 2^{m^2+m})$

The complexity of N_{k-2} is:

$$||N_{k-2}|| = O(2^{k-2} * m^k * 2^{m^2+m}) + ||N_{k-1}||$$

And so on. In general:

The complexity of N_{k-j} is:

$$||N_{k-j}|| = O(2^{k-j} * m^k * 2^{m^j+...+m}) + ||N_{k-j+1}||$$

Therefore, for the flat net N_0 (with $j = k$):

$$||N_0|| = O(m^k * 2^{\sum_{i=1}^{n} m^i}) + ||N_1|| \Rightarrow$$

$$||N_0|| = O(||N_k|| * 2^{c(k)*||N_k||})$$

where $c(k)$ is a function of k.

9.3 Flattening with non-atomic preemption

Other methods of flattening with silent transition insertion, more efficient in terms of the net complexity can be provided. For example, given a safe PCN one can build a polynomial equivalent PCN such that each place has a complementary place. Then, preemption is easy to implement using a BDD-net, i.e. a net which mimics a structure of the BDD representing a characteristic function of the corresponding ER. An alternative way of "almost atomic" preemption can be based on place freezing. In this method one empties all complementary pairs pair-by-pair such that after the first place pair has been emptied, all the others are freezed (no activity is allowed) and sooner or later all the pairs will be be emptied. This will trigger a preemptive transition.

In general it seems that exponential growth of the modeling PN is unavoidable. An informal argument is as follows: it is easy to build a PN inside a place chart which model an arbitrary Boolean function. Then, preempting all the places of this modeling net is similar to evaluation of the Boolean function. Since there is no polynomial representation which works for all Boolean functions, there is no polynomial equivalent PN.

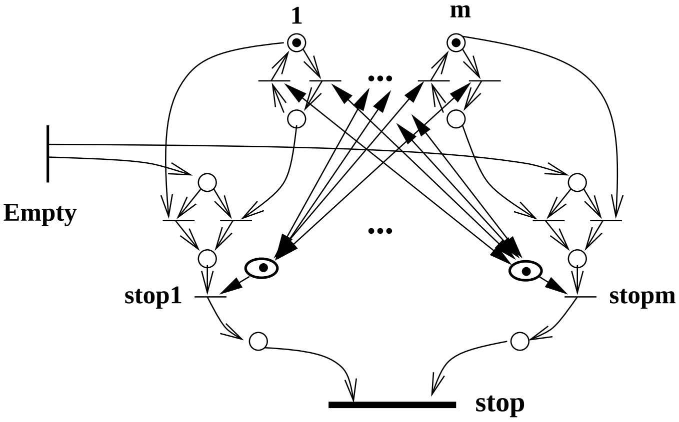


Figure 24: Flattening with non-atomic preemption

E.g., Figure 24 shows a PN weakly bisimilar to the PCN from Figure 22. Freezing arcs are shown with bold arrows. Contrary to the SOP-flattening these methods implement preemption as a *non-atomic* action. E.g., transition *stop*1 can preempt the first parallel process, while the others are simply blocked and we should wait until total preemption will occur. Therefore, the equivalence is weaker here. The complexity of removing one level of hierarchy for this example is polynomial $O(m * n)$, where m is a place count and n is a transition count. For k levels of hierarchy it is $O(2^k * m * k)$. Polynomial complexity for this example is due to a trivial enabling condition for a preemptive transition. In general an enabling condition requires exponential complexity.