Le pagine allegate sono tratte dal capitolo

Formal Languages and Concurrent Behaviours

Jetty Kleijn[1] and Maciej Koutny[2]

1) LIACS, Leiden University
P.O.Box 9512, NL-2300 RA Leiden, The Netherlands

2) School of Computing Science, Newcastle University
Newcastle upon Tyne, NE1 7RU, United Kingdom

Del volume:

Studies in Computational Intelligence , Volume 113, 2008

New Developments in Formal Languages and Applications
Editors: Gemma Bel-Enguix M. Dolores Jiménez-López, Carlos Martín-Vide
Springer.

Fact 14 : Let α be a trace.

- $\mathit{lin}(\mathit{canposet}(\alpha)) = \mathit{cantotalposet}(\alpha)$.[a]
- $\mathit{word}(\mathit{lin}(\mathit{canposet}(\alpha))) = \alpha$.

[a]Note that in $\mathit{cantotalposet}(\alpha)$ the trace α is treated as a set of words.

All information on the dependencies between the occurrences in a trace is represented in its uniquely associated poset.

5.3.1 Bibliographical Remarks

Main independent sources of trace theory are [7] (in the context of combinatorial problems) and [35, 29] (in the context of concurrency theory). An extensive account of trace theory is provided by [11] which, in particular, contains a chapter on dependence graphs [20]. For a bibliography on traces see [15].

5.4 Elementary Net Systems

In this section we first briefly discuss Petri nets as a system model, or rather as a framework for the modelling of concurrent systems. Then we introduce in more detail Elementary Net systems, the most basic Petri net model. In this model the key primitive notions underlying the operation of concurrent systems are explicitly represented and as such it has been the inspiration for the development of trace theory. In later sections, we will discuss more expressive net classes and how they lead to generalizations of traces.

The description of a Petri net comes in two parts, giving its static and dynamic aspects. The (static) structure of a Petri net is a graph specifying the local states (called places) of the system being modelled and its possible actions (called transitions). Global (system) states consist of combinations of the local states and it is the role of transitions to change those states in accordance with the given (dynamic) rules. Each transition has a neighbourhood of places with which it is linked and there are specific rules when transitions can occur (concurrently) and the effect of such occurrence. Both notions are fully determined by the transition's neighbourhood, i.e., every transition occurrence depends on neighbouring local states and also its effect when it occurs is completely local. A net system is fully specified when also an initial state is supplied from which possible behavioural scenarios are initiated. By varying the kind and nature of the relationships between places and transitions, as well as the precise notions of global state, and the enabling and occurrence rules, one obtains different classes of Petri nets.

First we introduce the basic structure underlying every Petri net. The definition below captures what presumably is the most fundamental class of nets.

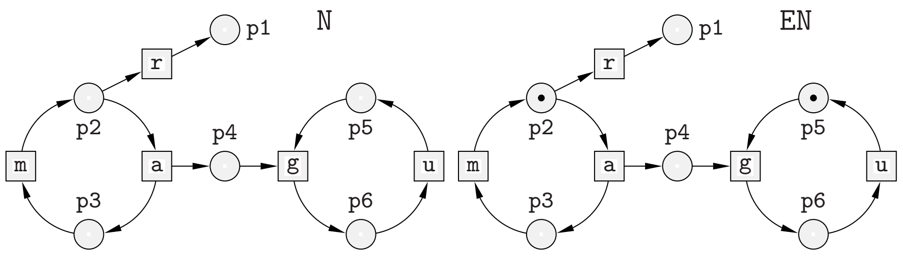

Fig. 5.6. A net without and with configuration (an EN-system) for the running example with a producer and a consumer subnet connected by a (buffer) place `p4`.

Definition 15 : nets

A net N is a relational tuple (P, T, F) with P and T disjoint finite sets of nodes, called respectively places and transitions, and $F \subseteq (T \times P) \cup (P \times T)$ the flow relation.

In diagrams, places are drawn as circles, and transitions as rectangles. The flow relation is represented by directed arcs between them. Hence nets are drawn as bipartite graphs.

Figure 5.6 shows the net $\mathtt{N} = (\mathtt{P}, \mathtt{T}, \mathtt{F})$, where $\mathtt{P} = \{\mathtt{p1}, \mathtt{p2}, \mathtt{p3}, \mathtt{p4}, \mathtt{p5}, \mathtt{p6}\}$ is the set of places, $\mathtt{T} = \{\mathtt{a}, \mathtt{g}, \mathtt{m}, \mathtt{r}, \mathtt{u}\}$ is the set of transitions, and the flow relation `F` comprises the following twelve arcs:

$$\begin{array}{llllll} (\mathtt{r},\mathtt{p1}) & (\mathtt{p2},\mathtt{r}) & (\mathtt{p3},\mathtt{m}) & (\mathtt{m},\mathtt{p2}) & (\mathtt{p2},\mathtt{a}) & (\mathtt{a},\mathtt{p3}) \\ (\mathtt{a},\mathtt{p4}) & (\mathtt{p4},\mathtt{g}) & (\mathtt{p5},\mathtt{g}) & (\mathtt{g},\mathtt{p6}) & (\mathtt{p6},\mathtt{u}) & (\mathtt{u},\mathtt{p5})\ . \end{array}$$

Let (P, T, F) be a net. The *inputs* and *outputs* of a node $x \in P \cup T$ are the sets ${}^\bullet x$ and $x^\bullet$, respectively comprising all y such that yFx and xFy, and the *neighbourhood* ${}^\bullet x^\bullet$ of x is the union of its inputs and outputs. The dot-notations readily extend to sets of nodes, e.g., ${}^\bullet X$ comprises all inputs of the nodes in X. It is assumed here that each net is *T-restricted* which means that every transition has at least one input (cause) and at least one output (effect). For the net `N` in Figure 5.6, ${}^\bullet\mathtt{g} = \{\mathtt{p4}, \mathtt{p5}\}$ and $\mathtt{p3}^\bullet = \{\mathtt{m}\}$.

5.4.1 Configurations and Transition Occurrence

In this and the next section, the states of a net $N \overset{\mathrm{df}}{=} (P, T, F)$ are given by subsets of places representing the conditions that hold at a given global situation.

Definition 16 : configurations

A configuration of a net is a subset of its places.

In diagrams, a configuration C is represented by drawing in each place p in C a token (a small black dot). A possible configuration for the net in Figure 5.6 is $\mathtt{C} = \{\mathtt{p2}, \mathtt{p5}\}$, as illustrated on the right of Figure 5.6.

Transitions represent actions which may occur at a given configuration and then lead to a new configuration.

Definition 17 : transition occurrences

A transition t can occur (or is enabled) at a configuration C if ${}^\bullet t \subseteq C$ and $t^\bullet \cap C = \varnothing$. Its occurrence then leads to a new configuration $(C \setminus {}^\bullet t) \cup t^\bullet$.

Thus a (potential) occurrence of a transition depends only on its neighbours. If t can occur at C then we write $C[t\rangle$, and if its occurrence leads to C' we write $C[t\rangle C'$. Note that through such an occurrence, all inputs of t cease to hold, and all outputs start to hold. Hence the change caused by the occurrence of a transition is always the same and does not depend on the current global state. For the configuration $\mathtt{C}$ shown in Figure 5.6, the enabled transitions are $\mathtt{r}$ and $\mathtt{a}$. Moreover, we have $\mathtt{C}[\mathtt{a}\rangle\{\mathtt{p3}, \mathtt{p4}, \mathtt{p5}\}$ and $\mathtt{C}[\mathtt{r}\rangle\{\mathtt{p1}, \mathtt{p5}\}$. Figure 5.7 provides further intuition about the enabling and occurrence rules for net transitions.

We now lift the execution of transitions to a concurrent context by allowing the simultaneous occurrence of transitions provided that they do not interfere with one another, i.e., their neighbourhoods are mutually disjoint.

Definition 18 : steps

A step of a net is a subset of its transitions. A step can occur (or is enabled) at a configuration C if the neighbourhoods of its transitions do not overlap, and each transition is enabled. The effect of its occurrence is the cumulative effect of the occurrences of the transitions it comprises.

In other words, a step U is enabled at C if ${}^\bullet t^\bullet \cap {}^\bullet t'^\bullet = \varnothing$ for all distinct transitions t and t' in U, and $C[t\rangle$ for each transition t in U. We denote this by $C[U\rangle$. The occurrence of an enabled step leads to a new configuration C' given by $(C \setminus {}^\bullet U) \cup U^\bullet$, and we denote this by $C[U\rangle C'$. Note that $C[U\rangle C$ *iff* the step U is empty $\checkmark$. For the configuration $\mathtt{C}$ shown in Figure 5.6, we have $\mathtt{C}[\{\mathtt{a}\}\rangle\mathtt{C}'$ where $\mathtt{C}' = \{\mathtt{p3}, \mathtt{p4}, \mathtt{p5}\}$; moreover, we further have:

$$\mathtt{C}'[\{\mathtt{m}, \mathtt{g}\}\rangle\{\mathtt{p2}, \mathtt{p6}\} \qquad \mathtt{C}'[\{\mathtt{m}\}\rangle\{\mathtt{p2}, \mathtt{p4}, \mathtt{p5}\} \qquad \mathtt{C}'[\{\mathtt{g}\}\rangle\{\mathtt{p3}, \mathtt{p6}\} \ .$$

We are now ready to introduce sequences of transitions and step occurrences.

Definition 19 : step sequences

A step sequence of a net is a finite sequence of non-empty steps occurring one after another from a given configuration.

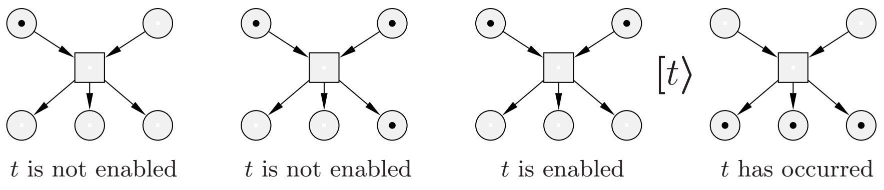

Fig. 5.7. Local change-of-state produced by the occurrence of a transition.

In other words, a step sequence from a configuration C to a configuration C' is a possibly empty sequence $\sigma = U_1 \dots U_n$ of non-empty steps U_i such that $C[U_1\rangle C_1, \dots, C_{n-1}[U_n\rangle C'$, for some configurations $C_1, \dots, C_{n-1}$. We also write $C[\sigma\rangle C'$ or $C[\sigma\rangle$, and say that C' is a configuration *reachable* from C. The set of all configurations reachable from C will be denoted by $[C\rangle$. Note that we always have $C \in [C\rangle$. If $n = 0$, thus $\sigma = \lambda$ the empty (step) sequence, then $C = C'$. The converse implication however does not hold $\checkmark$. For the configuration $\mathtt{C}$ shown in Figure 5.6, $\mathtt{C}[\{\mathtt{a}\}\{\mathtt{m},\mathtt{g}\}\{\mathtt{u},\mathtt{r}\}\rangle\{\mathtt{p1},\mathtt{p5}\}$, and, as we will see later on, the set $[\mathtt{C}\rangle$ comprises twelve reachable configurations.

To improve the readability of the notations when discussing examples, we will often drop the curly brackets when writing a singleton step, e.g., we can write $\mathtt{a}\{\mathtt{m},\mathtt{g}\}\mathtt{ur}$ instead of $\{\mathtt{a}\}\{\mathtt{m},\mathtt{g}\}\{\mathtt{u}\}\{\mathtt{r}\}$.

A special kind of step sequences are those that consist of singleton steps only. Such sequences (of transitions) are referred to as *firing sequences*. For example, $\mathtt{amgur}$ is a firing sequence from $\mathtt{C}$ to $\{\mathtt{p1},\mathtt{p5}\}$. Reachability of configurations does not depend on whether one uses step sequences or firing sequences. If, however, the structure of a net is enriched with inhibitor arcs as we will do it in the next section, then reachability may be affected by the restriction to firing sequences.

5.4.2 Concurrency and Causality

The definition of concurrent behaviour on basis of non-interference, as introduced above, allows one to investigate some intricate relationships in the way transitions can occur. As a first observation we have that transitions which can be executed simultaneously (at some configuration) do not have to occur together. They can still occur one after another. Moreover, whenever transitions can occur in any order, they must be concurrently enabled and non-interfering.

Fact 20 : Let C, C' be configurations and U, U' be steps of a net.

- $C[U \cup U'\rangle C'$ and $U \cap U' = \varnothing$ implies $C[UU'\rangle C'$.
- $C[UU'\rangle C'$ and $C[U'\rangle$ implies $U \cap U' = \varnothing$ and $C[U \cup U'\rangle C'$.

This fact is often referred to as a 'diamond property'. The reason is that if we have, say, $C[\{a,b\}\rangle C'$, it then follows that we also have $C[\{a\}\rangle C''[\{b\}\rangle C'$

and $C[\{b\}\rangle C'''[\{a\}\rangle C'$ where C'' and C''' are distinct configurations ✓. In drawing this yields a diamond shape. Note that the two statements together show that for the dynamics of nets defined sofar, diamonds imply concurrency and vice versa. For the configurations $\mathtt{C}' = \{\mathtt{p3}, \mathtt{p4}, \mathtt{p5}\}$ and $\mathtt{C}'' = \{\mathtt{p2}, \mathtt{p6}\}$ of the net shown in Figure 5.6, we have $\mathtt{C}'[\{\mathtt{m}, \mathtt{g}\}\rangle \mathtt{C}''$ as well as $\mathtt{C}'[\mathtt{mg}\rangle \mathtt{C}''$ and $\mathtt{C}'[\mathtt{gm}\rangle \mathtt{C}''$, and the resulting 'diamond' can be seen with a little bit of effort at the centre of the upper state graph in Figure 5.8.

The first part of Fact 20 implies that every step of simultaneously occurring transitions can be split into any partition of subsets occurring in sequence, with the same effect as the original step. As a consequence, every step sequence eventually gives rise to a valid (but not necessarily unique) firing sequence. And so the configurations reachable from a given one are the same for step sequences and firing sequences.

Fundamental relationships between transitions can be classified in a way which reflects their causal dependence (occurrence of one enables the other), competition for shared resources (both can occur, but they cannot occur together), or concurrency (they can occur together).

Definition 21 : fundamental situations - behavioural

Let t and t' be distinct transitions, and C be a configuration of a net.

- t causally depends on t' at C if $\neg C[t\rangle$ and $C[t't\rangle$.
- t and t' are in conflict at C if $C[t\rangle$, $C[t'\rangle$ and $\neg C[\{t, t'\}\rangle$.
- t and t' are concurrent at C if $C[\{t, t'\}\rangle$.

For the configuration $\mathtt{C}$ shown in Figure 5.6, we have that $\mathtt{g}$ causally depends on $\mathtt{a}$, and the latter is in conflict with $\mathtt{r}$. Moreover, $\mathtt{m}$ and $\mathtt{g}$ are concurrent at the configuration $\mathtt{C}' = \{\mathtt{p3}, \mathtt{p4}, \mathtt{p5}\}$.

It is interesting to note the difference between conflict and concurrency in terms of firing sequences: in case of conflict at a configuration, both are enabled to occur, but the occurrence of one disables the other, whereas in case of concurrency, the two transitions can occur in either order.

Fact 22 : Let t and t' be transitions, and C be a configuration of a net.

- If t causally depends on t' at C then $\neg C[tt'\rangle$ and $C[t't\rangle$.
- If t and t' are in conflict at C then $\neg C[tt'\rangle$ and $\neg C[t't\rangle$.
- If t and t' are concurrent at C then $C[tt'\rangle$ and $C[t't\rangle$.

These fundamental relationships between transitions are defined dynamically by referring to a global state. However, if two transitions are in one of these three relationships at some configuration, then none of the other relationships will ever hold for them (at whatever configuration) ✓. In fact, the (potential) relationships between transitions are determined by the graph structure.

Definition 23 : fundamental situations - structural

Let t and t' be two distinct transitions of a net N.

- t and t' are structurally causally related if ${}^\bullet t \cap t'^\bullet \neq \varnothing$ or $t^\bullet \cap {}^\bullet t' \neq \varnothing$.
- t and t' are in structural backward conflict if ${}^\bullet t \cap {}^\bullet t' \neq \varnothing$.
- t and t' are in structural forward conflict if $t^\bullet \cap t'^\bullet \neq \varnothing$.
- t and t' are structurally independent if ${}^\bullet t^\bullet \cap {}^\bullet t'^\bullet = \varnothing$.

For the net shown in Figure 5.6, `a` and `g` are structurally causally related, `a` and `r` are in structural forward conflict, and `r` and `u` are structurally independent.

5.4.3 EN-Systems and Their State Spaces

Having defined nets with states and dynamics, it is now time to study them as systems which start their operation from an initial state.

Definition 24 : EN-systems

An elementary net system (or EN-system) consists of an underlying net and an initial configuration. Its state space consists of all configurations reachable from the initial configuration.

In other words, an elementary net system EN is a relational tuple (P, T, F, C_{init}) such that the first three components form its underlying net and $C_{init} \subseteq P$ is the initial configuration. Figure 5.6 shows on the right an EN-system $\mathtt{EN} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{C_{init}})$, where $\mathtt{C_{init}} = \mathtt{C} = \{\mathtt{p2}, \mathtt{p5}\}$, modelling our running example. Its state space consists of twelve configurations:

$$[\mathtt{C_{init}}\rangle = \{\{\mathtt{pi}, \mathtt{pj}\} \mid \mathtt{i} = 1,2,3 \wedge \mathtt{j} = 5,6\} \cup \{\{\mathtt{pi}, \mathtt{p4}, \mathtt{pj}\} \mid \mathtt{i} = 1,2,3 \wedge \mathtt{j} = 5,6\} \; .$$

The *state graph* of EN is a relational tuple $stategr(EN) \overset{\text{df}}{=} ([C_{init}\rangle, LA, C_{init})$ with node set $[C_{init}\rangle$, set of labelled arcs $LA \overset{\text{df}}{=} \{(C, U, C') \mid C \in [C_{init}\rangle \wedge C[U\rangle C'\}$, and initial node C_{init}. Restricting the arcs of the state graph to those labelled by singletons steps yields the *sequential state graph* of EN, denoted by $seqstategr(EN)$. Figure 5.8 gives examples of each kind of state graph for the EN-system $\mathtt{EN_{simple}}$ in Figure 5.9.

Since every configuration reachable from the initial configuration by a step sequence is also reachable by a firing sequence, all nodes in $seqstategr(EN)$ are reachable from the initial node. Interestingly, also $stategr(EN)$ can be recovered from the sequential state graph $seqstategr(EN)$ by saturating the latter with non-singleton step labelled edges using the diamond property (Fact 20) ✓.

To illustrate the above idea, let us consider the state graphs in Figure 5.8, and two nodes, $\mathtt{C} = \{\mathtt{p3}, \mathtt{p4}, \mathtt{p6}\}$ and $\mathtt{C}' = \{\mathtt{p2}, \mathtt{p4}, \mathtt{p5}\}$. Looking at the sequential state graph, we can deduce that $\mathtt{C}[\{\mathtt{m}\}\{\mathtt{u}\}\rangle\mathtt{C}'$ and $\mathtt{C}[\{\mathtt{u}\}\rangle$. Hence, by the

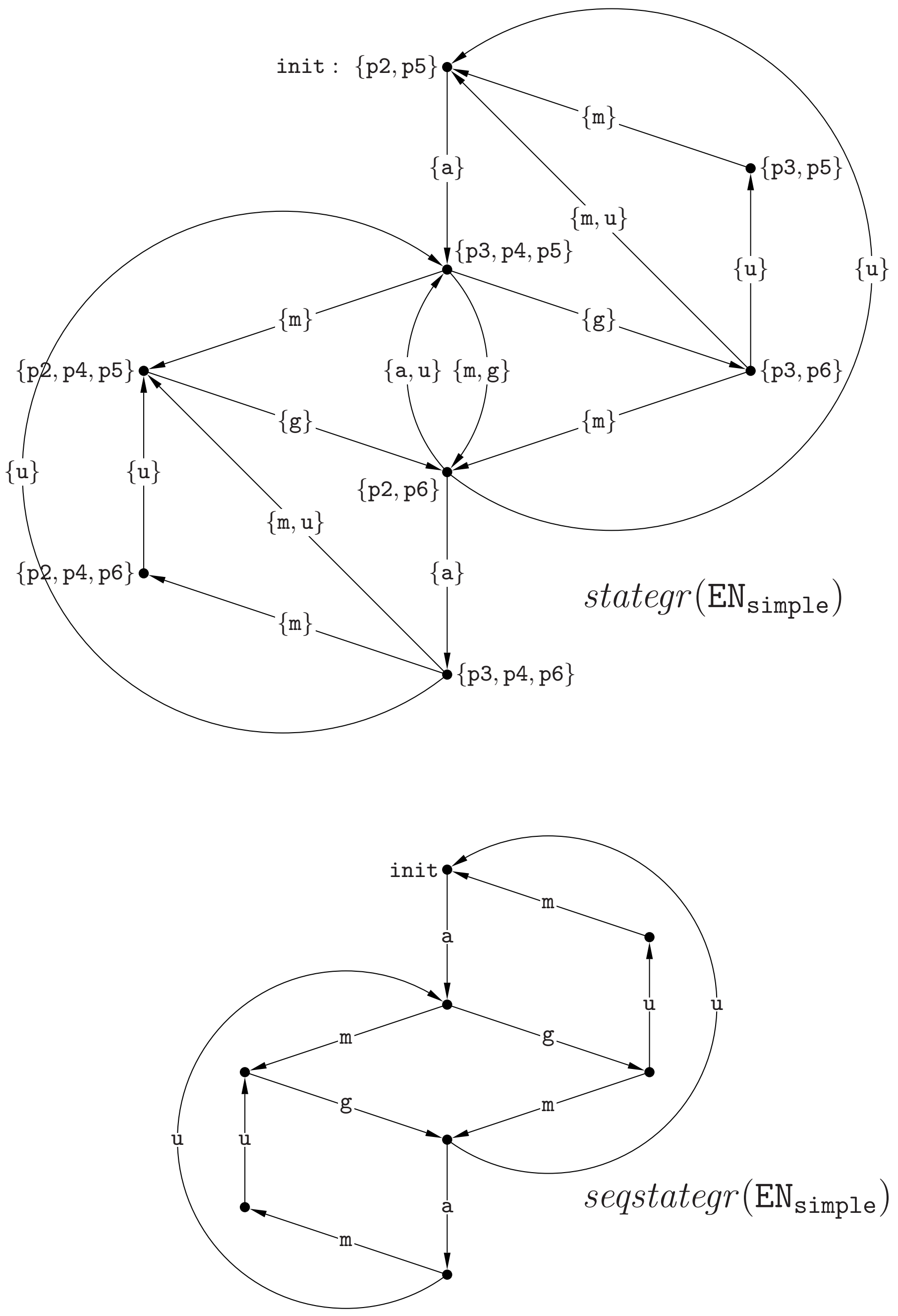

Fig. 5.8. The state graph of $\mathtt{EN}_{\mathtt{simple}}$ from Figure 5.9 and its sequential state graph.

second part of Fact 20, we have $\mathtt{C}[\{\mathtt{m},\mathtt{u}\}\rangle\mathtt{C}'$ and so the concurrent step $\{\mathtt{m},\mathtt{u}\}$ from $\mathtt{C}$ to $\mathtt{C}'$ in the state graph has been deduced from purely sequential information.

For a behavioural comparison of EN-systems, isomorphism is too discriminating, because then there would be essentially only one structure defining the behaviour under consideration. Therefore, in EN-system theory it is the state graph which provides the main reference point for any behaviour related analysis. However, all information on (the relevant, active, part of) the net underlying the EN-system can still be recovered from its state graph; the places belonging to reachable configurations, transitions which actually occur and thus appear in the steps labelling the arcs, and their neighbourhood relations, are all explicitly represented in the state graph. Using the state graph itself would thus lead to a similar identification of net structure and behaviour. To abstract from the concrete information on places and transitions, state graph isomorphism is used as an equivalence notion for the comparison of concurrent behaviours. Already the structure of its state graph provides a complete and faithful representation of the behaviour of an EN-system. In particular, causality, conflict, and concurrency among (possibly renamed) transitions can be determined from it. Note that two EN-systems have isomorphic state graphs *iff* also their sequential state graphs are isomorphic $\checkmark$. After isomorphism of EN-systems, state graph isomorphism is the second strongest notion of equivalence employed in the behavioural analysis of EN-systems. With this equivalence it is possible to transform EN-systems in order to realise a desired property or feature (a normal form) without affecting their dynamic properties in an essential way, i.e., the state graph remains the same up to isomorphism and the resulting system is considered behaviourally equivalent. An important application of this idea is the following.
The enabling relation for transitions checks explicitly for the emptiness of their output places. This may be regarded as somewhat unsatisfactory. It would be more efficient and intuitively more appealing if it would be sufficient to check only whether all input conditions are fulfilled.

Definition 25 : contact-freeness

An EN-system is contact-free if for every reachable configuration C and every transition t, it is the case that ${}^\bullet t \subseteq C$ implies $t^\bullet \cap C = \varnothing$.

In other words, a contact-free system is one where the test for transition enabledness can simply be ${}^\bullet t \subseteq C$ without changing anything. The EN-system shown in Figure 5.6 is <u>not</u> contact-free $\checkmark$. Not all EN-systems are contact-free, but the simple transformation described next turns any EN-system into a behaviourally equivalent contact-free version.

Two places, p and q, are *complements* of one another if ${}^\bullet p = q^\bullet$, $p^\bullet = {}^\bullet q$ and exactly one of them belongs to the initial configuration C_{init}. The *complementation* $\widetilde{EN}$ of *EN* is obtained by adding, for each place p without a

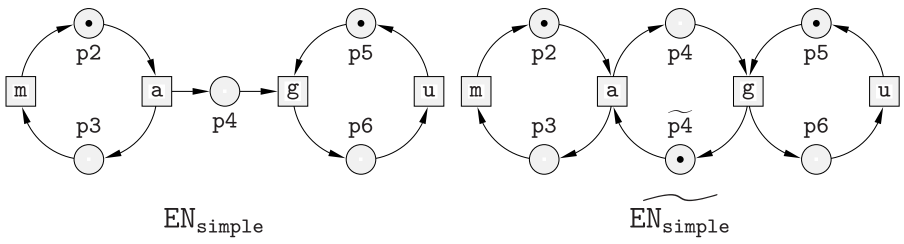

Fig. 5.9. A simplified version of the EN-system from Figure 5.6 and its complementation.

complement, a fresh complement place $\widetilde{p}$; moreover, if the initial configuration does not contain p then $\widetilde{p}$ is added there as well. The result is clearly an EN-system and the two systems have isomorphic state spaces. In fact, only the reachable configurations have to be renamed in the case that new complement places have been added; the arc labels between corresponding states however are the same ✓.

Fact 26 : $\widetilde{EN}$ is contact-free and its state space is isomorphic to that of EN.

The construction is illustrated by the non-contact-free EN-system $\mathtt{EN_{simple}}$ in Figure 5.9 and its contact-free complementation $\widetilde{\mathtt{EN_{simple}}}$. The state spaces of the two EN-systems are respectively:

$$\begin{array}{ll} \mathtt{Conf} \cup \{\{\mathtt{pi},\mathtt{pj}\} \mid \mathtt{i}=2,3 \wedge \mathtt{j}=5,6\} & \text{(left)} \\ \mathtt{Conf} \cup \{\{\mathtt{pi},\widetilde{\mathtt{p4}},\mathtt{pj}\} \mid \mathtt{i}=2,3 \wedge \mathtt{j}=5,6\} & \text{(right)} \end{array}$$

where $\mathtt{Conf} = \{\{\mathtt{pi},\mathtt{p4},\mathtt{pj}\} \mid \mathtt{i} = 2,3 \wedge \mathtt{j} = 5,6\}$. It is can be seen that a suitable isomorphism for their state graphs maps each $\{\mathtt{pi},\mathtt{pj}\}$ to $\{\mathtt{pi},\widetilde{\mathtt{p4}},\mathtt{pj}\}$, and is the identity for the configurations in `Conf`.

Fact 26 assumes that one adds complements for all non-complemented places. But it is also possible to add complementation selectively and, in general, we have that any EN-system with an arbitrary, added set of new complement places has a state space which is isomorphic to that of the original EN-system ✓. For the EN-system `EN` modelling the running example we can add a complement of the buffer place which results in the equivalent EN-system shown in Figure 5.10. In this case already the selective complementation yields a contact-free EN-system ✓.

Since it is always possible to ensure contact-freeness without changing the behaviour represented in the state-graph, we now make a simplifying assumption.

In the rest of this tutorial all EN-systems are **contact-free**.

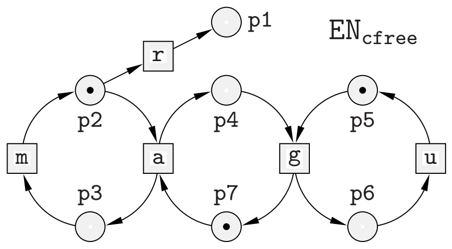

Fig. 5.10. A contact-free version of the EN-system from Figure 5.6 where the place p4 has been complemented, i.e., $\mathtt{p7} = \widetilde{\mathtt{p4}}$.

5.4.4 Behaviour of EN-Systems

Let $EN = (P, T, F, C_{init})$ be a fixed EN-system for the rest of this section.

In addition to the state graph, we can also associate firing sequences and step sequences as behavioural notions to EN-systems. The set of all firing sequences $\mathit{firseq}(EN)$ of EN consists of those sequences $u \in T^*$ such that $C_{init}[u\rangle$ and, similarly, the set of all step sequences $\mathit{stepseq}(EN)$ of EN comprises all step sequences of EN from C_{init}. Each firing sequence corresponds to a finite labelled path through the sequential state graph from the initial node. Since the set of reachable configurations of an EN-system is finite, the sequential state graph is a finite state machine. Hence the set of firing sequences of an EN-system is a prefix-closed regular language. However, it consists of purely sequential observations of the EN-system's behaviour without any reference to the possible independence of transitions. Yet such causality information is often of high importance for system analysis and design.

Let us first demonstrate how the theory of traces can be applied to extract partial orders from firing sequences as representations of the *necessary* causal ordering of transition occurrences within these sequences.

Definition 27 : concurrency alphabets of EN-systems

The concurrency alphabet of EN is $CA_{EN} \overset{\mathrm{df}}{=} (T, Ind_{EN})$ where the structural independence relation Ind_{EN} comprises all pairs of distinct transitions with disjoint neighbourhoods.

Defined in this way, $Ind_{EN} = \{(t, t') \mid t, t' \in T \wedge {}^\bullet t^\bullet \cap {}^\bullet t'^\bullet = \varnothing\}$ is a symmetric and irreflexive relation and so it is indeed an independence relation. For the EN-system $\mathtt{EN_{cfree}}$ in Figure 5.10, $\mathtt{Ind_{EN_{cfree}}} = \mathtt{Ind}$ where $\mathtt{Ind}$ was defined at the beginning of Section 5.3. An important observation is now that in a firing sequence adjacent occurrences of independent transitions could have occurred also in the other order (see the diamond property, Fact 20). Hence, for every firing sequence of EN, all its trace equivalent words from T^* are also firing sequences of EN.

Fact 28 : $\mathit{firseq}(EN) = \bigcup_{u \in \mathit{firseq}(EN)} [u]$.

Taking, for example, $\mathtt{EN}_{\mathtt{cfree}}$ in Figure 5.10, we have $\mathtt{agm} \in \mathit{firseq}(\mathtt{EN}_{\mathtt{cfree}})$ and $[\mathtt{agm}] = \{\mathtt{agm}, \mathtt{amg}\}$. Clearly, $\mathtt{amg}$ is also a firing sequence of $\mathtt{EN}_{\mathtt{cfree}}$.

The step sequences of an EN-system obviously provide important insights into concurrency aspects of its behaviour. They are nevertheless still sequential rather than concurrent in nature in the sense that the sequential ordering of the steps obscures the true causal dependencies between the occurrences of transitions. Petri net models can however easily support a formal approach where this information is readily available by unfolding behaviours into structures allowing an explicit representation of causality and concurrency.

5.4.5 Non-Sequential Observations

Rather than describing the behaviour of the system in terms of sequential observations, like firing sequences and step sequences, we now present a semantics based on a class of acyclic Petri nets, called *occurrence nets*. What one essentially tries to achieve here is to record the changes of configurations due to transitions being executed along some legal behaviour of the EN-system, and in doing so record which places were emptied (served as inputs) and which filled (as outputs). The resulting occurrence nets may be viewed as partial net unfoldings, with each transition representing an occurrence of a transition in the original net (thus occurrence nets are acyclic), and each place corresponding to the occurrence of a token on a place of the original net. Conflicts between transitions are resolved and thus the places in an occurrence net do not branch.

Definition 29 : occurrence nets

An occurrence net is a relational tuple $ON \overset{\mathrm{df}}{=} (B, E, R, \ell)$ such that (B, E, R) is an underlying net,[a] ℓ is a labelling for $B \cup E$, R is an acyclic flow relation, and $|{}^\bullet b| \leq 1$ and $|b^\bullet| \leq 1$, for every $b \in B$.

[a]The dot-notations, configurations, firing rule, etc, for ON are as those defined for the underlying net.

The places of an occurrence net are usually called *conditions* (‘Bedingungen’ in German) and its transitions *events* (‘Ereignisse’ in German). The default *initial configuration* of ON consists of all conditions without incoming arcs, i.e., C_{init}^{ON} comprises all conditions $b \in B$ such that ${}^\bullet b = \varnothing$, and the default

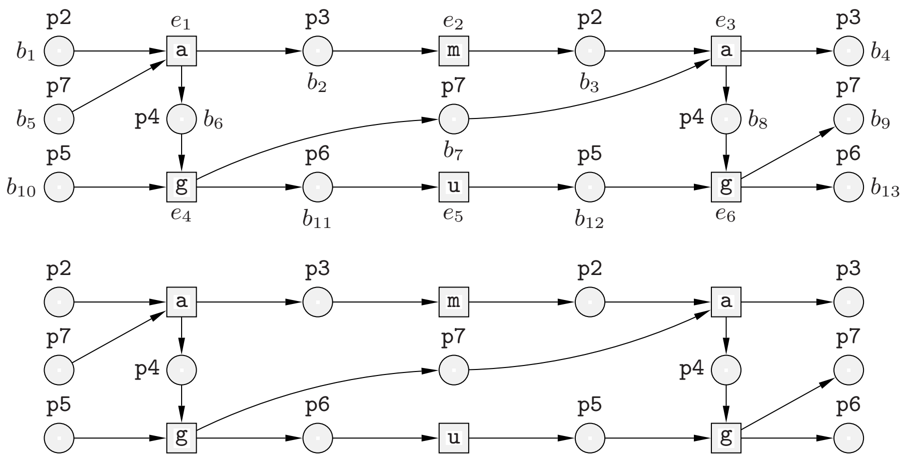

Fig. 5.11. An occurrence net `ON` with nodes labelled by places and transitions of the EN-system $\mathtt{EN_{cfree}}$ in Figure 5.10 (top), and the same occurrence net with identities of the nodes omitted (bottom).

final configuration C^{ON}_{fin} consists of all conditions without outgoing arcs. The default initial configuration of the occurrence net in Figure 5.11 is $C^{\mathtt{ON}}_{init} = \{b_1, b_5, b_{10}\}$ and the default final configuration is $C^{\mathtt{ON}}_{fin} = \{b_4, b_9, b_{13}\}$.

The sets of firing and step sequences of ON are defined w.r.t. the default initial configuration. However, since an occurrence net is meant to represent a record of a concurrent run of an EN-system, what really counts is not the identities of its events, but their labels which are linked to the occurrences of transitions in the EN-system. The *language* of ON is the set $language(ON)$ of all sequences $\ell(u)$ such that u is a firing sequence from the default initial configuration of ON to the default final configuration.

By abstracting from the conditions we associate with the occurrence net $ON = (B, E, R, \ell)$ a directed acyclic graph with E as its set of nodes. This dag $dag(ON) \overset{\mathrm{df}}{=} (E, R \circ R|_{E\times E}, \ell|_E)$ represents the direct causal relationships between the events. Its transitive closure $dag(ON)^+$, see Figure 5.12, then gives all, direct and indirect, causal dependencies. For example, e_4 directly causes e_5, but there is only an indirect causal link from e_4 to e_6.

ON with its default initial configuration is basically a contact-free EN-system ✓. Interestingly, all the sets occurring in any step sequence σ from the initial configuration to another configuration C, are mutually disjoint ✓. Moreover, C is the default final configuration *iff* the steps in σ use all the events of the occurrence net ✓.

A *slice* of ON is a maximal (w.r.t. set inclusion) subset S of events from ON which are causally unrelated, i.e., $(S \times S) \cap R^+ = \varnothing$. The set of all slices of ON is denoted by $slices(ON)$. Clearly, both default configurations are slices and, in general, $[C^{ON}_{init}\rangle = slices(ON)$, i.e., slices are exactly those configurations which are reachable from the initial configuration ✓. Moreover, the

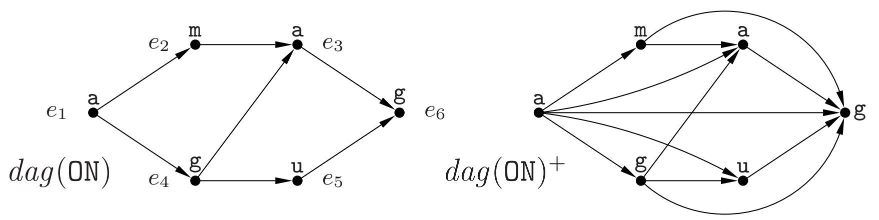

Fig. 5.12. Direct causality among the events in the occurrence net in Figure 5.11, and full causality (node identities omitted).

final configuration of *ON* is always reachable from any configuration reachable from the initial one $\checkmark$. Essentially, this means that *ON* is deadlock-free until its final configuration has been reached.

The processes of an EN-system are occurrence nets reflecting its structure and possible behaviour through their labelling and initial configuration.

Definition 30 : processes of EN-systems

A process of *EN* is an occurrence net $ON = (B, E, R, \ell)$ such that:

- ℓ labels conditions with places and events with transitions.
- ℓ is injective on the default initial configuration of *ON*, as well as on the sets of input and output conditions of each event.
- $\ell(C^{ON}_{init}) = C_{init}$ and, for every $e \in E$, $\ell(^\bullet e) = {}^\bullet\ell(e)$ and $\ell(e^\bullet) = \ell(e)^\bullet$.

The occurrence net ON in Figure 5.11 is a process of the EN-system in Figure 5.10.

Processes can be used to investigate the behaviours of EN-systems. Due to the second and third conditions in Definition 30, we can relate the firing sequences, step sequences and configurations of *EN* to their labelled versions in *ON*. More precisely, if we take a step sequence $C^{ON}_{init}[\sigma\rangle C$ then $C_{init}[\ell(\sigma)\rangle\ell(C)$ holds. This can be proved by an inductive argument from which it also follows that labelling of *ON* is injective on all its slices and hence also on the sets occurring in any step sequence of *ON* $\checkmark$. If σ is a step sequence from the default initial configuration of *ON*, then $\ell(\sigma)$ is referred to as a *labelled step sequence* of *ON*. Similar to the language of *ON*, the *step language* of *ON* is defined as the set *steplanguage*(*ON*) of all sequences $\ell(\sigma)$ such that σ is a step sequence from the default initial configuration of *ON* to the default final configuration.

In general, it follows that all firing and step sequences of EN-systems can be derived from their processes.

Fact 31 : Let $\mathcal{ON}$ be the set of all processes of EN.

- $\mathit{firseq}(EN) = \bigcup_{ON \in \mathcal{ON}} \mathit{language}(ON)$.
- $\mathit{stepseq}(EN) = \bigcup_{ON \in \mathcal{ON}} \mathit{steplanguage}(ON)$.

Definition 30 does not provide any clues as to how to derive a process of an EN-system. This is rectified in the next definition which shows how to construct a process corresponding to a given step sequence.

Definition 32 : processes construction

The occurrence net ON_σ generated by a step sequence $\sigma = U_1 \dots U_n$ of EN is the last element in the sequence $N_0, \dots, N_n$ where each N_k is an occurrence net (B_k, E_k, R_k, ℓ_k) constructed thus.
Step 0: $B_0 \overset{\text{df}}{=} \{p^1 \mid p \in C_{init}\}$ and $E_0 = R_0 \overset{\text{df}}{=} \varnothing$.
Step k: Given N_{k-1} we extend the sets of nodes and arcs as follows:

$$\begin{aligned} B_k &\overset{\text{df}}{=} B_{k-1} \cup \{p^{1+\triangle p} \mid p \in U_k^\bullet\} \\ E_k &\overset{\text{df}}{=} E_{k-1} \cup \{t^{1+\triangle t} \mid t \in U_k\} \\ R_k &\overset{\text{df}}{=} R_{k-1} \cup \{(p^{\triangle p}, t^{1+\triangle t}) \mid t \in U_k \wedge p \in {}^\bullet t\} \\ &\qquad\quad \cup \{(t^{1+\triangle t}, p^{1+\triangle p}) \mid t \in U_k \wedge p \in t^\bullet\} \ . \end{aligned}$$

In the above, the label of each node x^i is set to be x, and $\triangle x$ denotes the number of nodes of N_{k-1} labelled by x.

The construction is illustrated in Figure 5.13 for the $\mathtt{EN_{cfree}}$ in Figure 5.10 and its step sequence $\sigma = \mathtt{a}\{\mathtt{m},\mathtt{g}\}\{\mathtt{a},\mathtt{u}\}\mathtt{g}$. The resulting occurrence net is isomorphic to the occurrence net $\mathtt{ON}$ in Figure 5.11 which is a process of $\mathtt{EN_{cfree}}$.

Fact 33 : Each occurrence net constructed as in Definition 32 is a process of EN and, for each process of EN, there is a run of the construction from Definition 32 generating an isomorphic occurrence net.

Thus the operationally defined processes and the axiomatically defined processes of an EN-system are essentially the same.

Finally, we return to the trace semantics of EN-systems in relation to processes. First note that each trace gives rise to only one process, since interchanging adjacent occurrences of independent transitions has no effect on the construction of a process. So, $ON_u = ON_w$ whenever u and w are trace equivalent firing sequences ✓. Hence $ON_{[u]}$ the process associated to a trace is a well-defined notion. Conversely, the language of a process is identical to its defining trace ✓. Thus we have a one-to-one correspondence between traces and the processes of an EN-system. Moreover, even though the dag

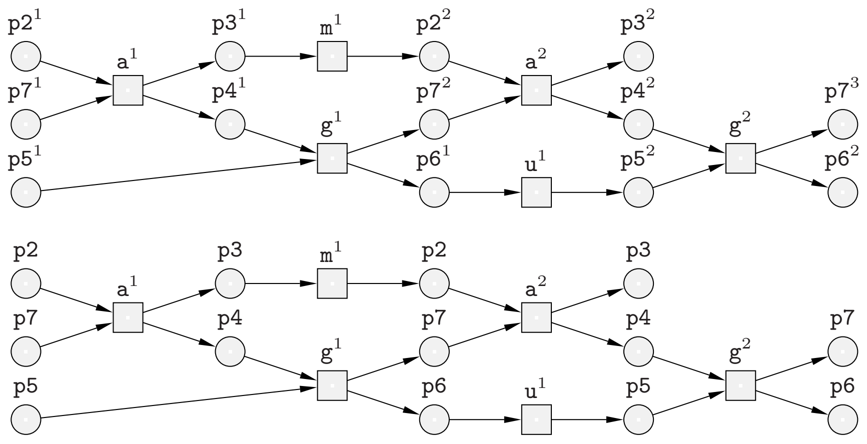

Fig. 5.13. The occurrence net $\mathtt{ON}_{\mathtt{a\{m,g\}\{a,u\}g}}$ generated for the EN-system in Figure 5.10: node-oriented view (top), and label-oriented view (bottom).

defined by a process is not necessarily isomorphic to the dependence graph of its trace ✓, they always define the same partial order on their transition occurrences.

Fact 34 : Let u be a firing sequence of EN.

- $[u] = language(ON_u)$.
- $canposet([u]) = dag(ON_u)^+$.

To conclude, the trace semantics and the process semantics of EN-systems lead to one partial order semantics by providing for each EN-system the same (isomorphic) partial orders modelling the causalities in its concurrent executions. This provides a strong argument in favour of the view that both these approaches capture the essence of causality in the behaviours of EN-systems.

5.4.6 Bibliographical Remarks

Over the past 40 or so years different classes of Petri nets have been introduced by varying the kind of underlying net, notion of local state, or transition relation. An early systematic treatment of basic notions in net theory and EN-systems can be found in [48]. Other extensions of the EN-systems approach adopt notions like priorities, real-time behaviour, or object-orientation. (In fact, we consider two such extensions later in this tutorial.) The general question of the intrinsic or common properties of nets is discussed in [8]. The problem of associating non-sequential semantics with Petri nets is dealt with, in particular, in [35, 40, 36, 37, 41, 42, 19, 47]. There is a systematic way of dealing with process semantics of various classes of Petri nets proposed in [31] which makes it possible to separately discuss behaviour, processes, causality,

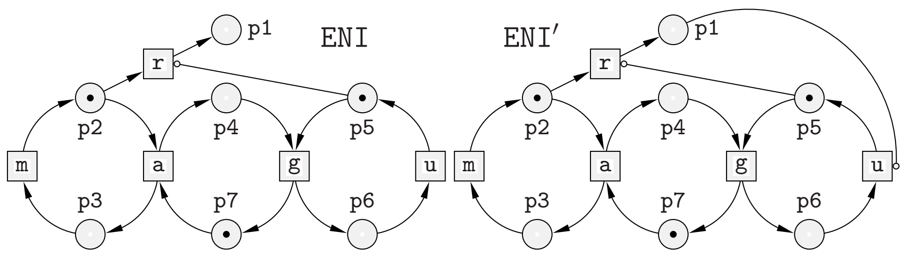

Fig. 5.14. Two ENI-systems modelling two variations of the running example.

and their mutually consistency. General Petri net related resources can be found in the web pages at [26].

5.5 Adding Inhibitor Arcs

This section extends the treatment of concurrency considered so far in EN-systems in order to accommodate the practically relevant case of nets with inhibitor arcs. In particular, we will demonstrate how the original definition of traces may be extended to describe in an adequate way also the additional features of the resulting new kind of concurrent behaviours.

To see why inhibitor arcs can be a convenient modelling device, let us imagine that a designer would like to modify the running example so that the producer cannot retire if the customer is waiting for an item. Such a modification is easily achieved by taking the EN-system of Figure 5.10 and adding to it an inhibitor arc linking the place `p5` and transition `r`. This yields the net system `ENI` shown on the left of Figure 5.14. (Inhibitor arcs are drawn with small open circles as arrowheads.) Adding this arc means that `r` cannot be enabled if `p5` contains a token, and so the producer indeed cannot retire if the consumer is waiting for an item. Elementary net systems with inhibitor arcs, or simply ENI-systems, thus extend EN-systems. The usefulness of inhibitor arcs stems from their ability to detect a lack rather than the presence of specific resources, i.e., tokens in specific places. That such an addition to the EN-system syntax is a true extension of their modelling power follows from the observation that there is no EN-system with exactly the same set of firing sequences as `ENI`. This can be shown by considering two firing sequences of `ENI`, `amgru` and `amgu`. If there was an EN-system generating the same firing sequences as `ENI`, then, due to the second statement in Fact 20, it would also have to generate the firing sequence `amgur`. But such a firing sequence is not generated by `ENI` as executing the last transition would contradict the defining characteristic of the inhibitor arc between `r` and `p5`. We will return to this example after introducing ENI-systems more formally.