Eric Badouel, Philippe Darondeau

Petri Net Synthesis

SPIN Springer's internal project number, if known

– Monograph –

December 2, 2011

Springer

Contents

Introduction

Part I

Elementary Net Synthesis

1

Introduction to Elementary Net Synthesis

This chapter aims at giving, for the particular class of elementary net systems, a first account of the principles of region based net synthesis. A brief introduction to elementary net systems (sections 1.1 and 1.2) provides the material needed in this book. We present in section 1.3 the construction of the saturated net version of a transition system, given by Ehrenfeucht and Rozenberg, and we explain in section 1.4 their characterization by the regional axioms of the net realizable transition systems. In fact, we do not confine our attention uniquely to elementary net systems but we consider also a slight extension thereof, enabling us to construct in section 1.5 an order-theoretic connection between transition systems and net systems. This order-theoretic connection provides a base for studying approximate realizations of transition systems by net systems. We vary subsequently the goals of net synthesis by considering first in section 1.6 the synthesis of net systems from transition systems with potential confusions of states, and next in section 1.7 the synthesis of net systems from regular languages. We finally explain in section 1.8 the context of Labelled Partial 2-Structures in which regions were discovered and first applied to elementary net synthesis. The chapter ends with a series of exercices.

1.1 An Informal Introduction to Elementary Nets

Elementary nets are a model of dynamic systems whose phase space and trajectories can be described in terms of a finite set of boolean properties. A state is characterized by stipulating for each property whether it holds or does not hold in that state. A transition is characterized by stipulating for each property whether it is turned valid, or it is turned invalid, or it is not affected by that transition. A specific aspect of dynamic systems modelled by elementary nets is to enable possibly several transitions from a given state. In an elementary net, properties are represented by places that may be marked or unmarked according to whether these properties hold or do not hold. By convention, places are drawn as circles, containing one token when they are marked and no token otherwise, and transitions are represented as squares. An incoming arc from a place to a transition means that the transition cannot occur unless that place is marked with a token which the transition consumes. An outgoing arc from a transition to a place means that the transition produces a token in that place and cannot occur if that place is already marked.

For the sake of an illustration, let us design an elementary net model for the famous dining philosophers problem [26], or more precisely for a reduction of this problem to three philosophers. Let us recall the statement of the problem. Three philosophers φ_1, φ_2, and φ_3 are sitting at a table with a bowl of spaghetti in the center. Three forks f_1, f_2, f_3 are placed on the table, such that philosopher φ_i has the fork f_i on his right and the fork $f_{i+1 \bmod 3}$ on his left (see Fig. 1.1). A philosopher alternates periods of eating and periods of thinking. To eat, he needs both the fork to his left and the fork to his right. Therefore, he tries to grab them one after the other while thinking. A philosopher who thinks with a fork in each hand stops thinking and starts eating after a finite delay. A philosopher who eats eventually stops eating, puts down the forks, and starts thinking again after a finite delay. The basic problem is to avoid deadlock, i.e., the situation in which every philosopher has taken one fork. A classical solution is to let all philosophers but one, say φ_1, take first the fork to their right, and to let φ_1 take first the fork to his left. An augmented problem is to avoid the starvation of any philosopher.

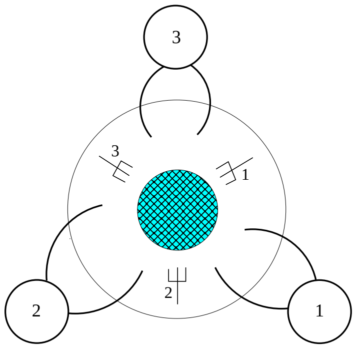


Fig. 1.1. modelling three Dining Philosophers

In order to ease the design, consider first a single philosopher, say φ_1 with fork f_1 available on his right and fork f_2 available on his left. The behaviour of philosopher φ_1 may be described in terms of five properties, two pertaining to fork f_1, two pertaining to fork f_2, and one pertaining to both forks. Fork f_1 may be **free** (property $f1f$), or philosopher φ_1 may **hold** that fork without eating (property $1h1$), or philosopher φ_1 may hold that fork and be **eating** (property $1e$). These properties are mutually exclusive, and if we let $f2f$ and $1h2$ be the counterparts of $f1f$ and $1h1$ for fork f_2, then similarly, $f2f$, $1h2$

and $1e$ are mutually exclusive. When fork f_1 is free, philosopher φ_1 can **take** it (transition $1t1$), thus invalidating property $f1f$ and validating property $1h1$. When fork f_2 is free, philosopher φ_1 can **take** it (transition $1t2$), thus invalidating property $f2f$ and validating property $1h2$. As a philosopher has two hands, these two events may occur concurrently, but we will assume that they cannot occur simultaneously. When philosopher φ_1 holds the two forks, he can **start eating** (transition $1se$), thus invalidating properties $1h1, 1h2$ and validating property $1e$. When philosopher φ_1 is eating, he can **start thinking** and put down the forks, thus invalidating property $1e$ and validating properties $f1f$ and $f2f$. These intuitions are captured in the elementary net system shown in Fig. 1.2.

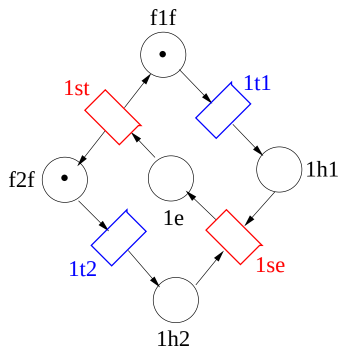


Fig. 1.2. elementary net model N_1 for Philosopher φ_1

The behaviour of this elementary net is described by the initialized transition system shown in Fig. 1.3. States are sets of marked places representing valid properties. Labels of arcs are transitions of the net. In the initial state (figured by a wriggling arrow), the two concurrent transitions $1t1$ and $1t2$ form a diamond, i.e., they may be fired one after another in any order and this order is not reflected in the resulting state.

$$\begin{tikzcd}
 & \{f1f,f2f\} \arrow[dl, "1t2"'] \arrow[dr, "1t1"] & & \\
\{f1f,1h2\} \arrow[dr, "1t1"'] & & \{1h1,f2f\} \arrow[dl, "1t2"] & 1e \arrow[ull, bend right, "1st"'] \\
 & \{1h1,1h2\} \arrow[urr, bend right, "1se"'] & &
\end{tikzcd}$$

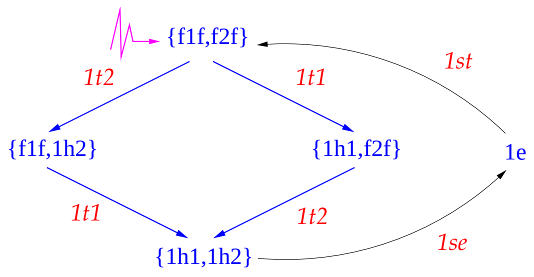


Fig. 1.3. behaviour of the elementary net for Philosopher φ_1

In order to construct an elementary net model for three dining philosophers, it suffices now to consider elementary net models N_1, N_2, N_3 for the respective philosophers $\varphi_1, \varphi_2, \varphi_3$ and to glue them on places $f1f, f2f, f3f$ as indicated in Fig. 1.4.

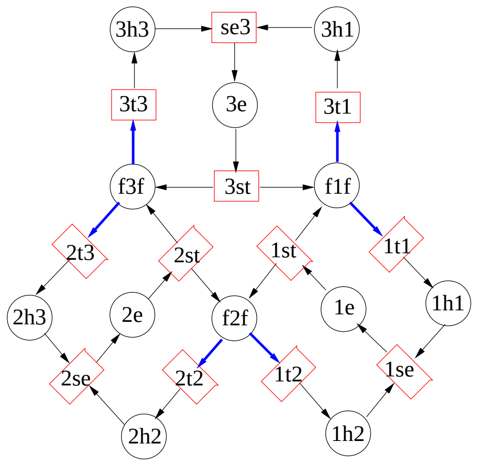


Fig. 1.4. elementary net model $N_{1,2,3}$ for three dining philosophers

Note the presence of conflicts in the elementary net $N_{1,2,3}$. Indeed, each one of places $f1f, f2f, f3f$ has two outgoing arcs, indicating that two philosophers compete for grabbing the corresponding fork. E.g., when fork f_1 is free (property $f1f$), it may be taken by philosopher φ_1 (transition $1t1$) or by philosopher φ_3 (transition $3t1$). The behaviour of the elementary net $N_{1,2,3}$ is described by an initialized transition system with 36 states and 78 arcs, and it is too large to be depicted in a drawing. Therefore, we show it as a list of labelled transitions between states numbered from 0 to 35. The initial state is numbered 0. Each transition (i, ℓ, j) in the list represents an arc from state i to state j labelled with ℓ. The list of transitions is the following.

(0,3t1,1)	(2,2t2,7)	(6,2t3,8)	(13,1t1,14)	(19,2t2,10)	(24,3t3,21)
(0,3t3,19)	(2,1t2,21)	(7,3se,4)	(14,2st,15)	(19,1t2,20)	(24,2t3,25)
(0,2t3,29)	(3,3st,0)	(8,2se,9)	(15,3t3,16)	(19,1t1,16)	(26,3t1,25)
(0,2t2,5)	(3,2t2,4)	(9,2st,1)	(15,2t3,31)	(20,3t1,21)	(26,1t1,27)
(0,1t2,23)	(3,1t2,22)	(10,3t1,7)	(15,2t2,35)	(20,1t1,17)	(27,1se,28)
(0,1t1,15)	(4,3st,5)	(10,1t1,11)	(15,1t2,33)	(21,3se,22)	(28,1st,29)
(1,3t3,2)	(5,3t1,6)	(12,2se,13)	(16,2t2,11)	(22,3st,23)	(29,3t1,30)
(1,2t3,30)	(5,3t3,10)	(12,3t1,8)	(16,1t2,17)	(23,3t1,24)	(29,2t2,12)
(1,2t2,6)	(5,2t3,12)	(12,1t1,32)	(17,1se,18)	(23,3t3,20)	(29,1t2,26)
(1,1t2,24)	(5,1t1,35)	(13,2st,0)	(18,1st,19)	(23,2t3,26)	(29,1t1,31)
(2,3se,3)	(6,3t3,7)	(13,3t1,9)	(19,3t1,2)	(23,1t1,33)	(30,2t2,8)

(30,1t2,25)	(32,2se,14)	(33,3t3,17)	(34,3t3,18)	(35,2t3,32)
(31,2t2,32)	(33,1se,34)	(33,2t3,27)	(34,2t3,28)	
(31,1t2,27)		(34,1st,0)	(35,3t3,11)	

This example shows that elementary net systems may be much more compact than their behaviours and it illustrates what is usually called the combinatorial explosion due to concurrency. The role of elementary net synthesis, which will be introduced in this chapter, is to reverse the process that leads from an elementary net system to an initialized transition system, thus providing more compact representations and exhibiting implicit concurrency.

1.2 Elementary Net Systems and their Firing Rule

An elementary net is defined over a set of *places* P. Places represent observable properties that may hold or not hold in a given state. A state or *marking* is a map $M : P \to \{0,1\}$, namely the characteristic function of the set of properties valid in that state. A place $p \in P$ is *marked* in M if $M(p) = 1$, unmarked otherwise. A marking may be seen equivalently as a set of marked places, hence as a subset of P. This is the interpretation adopted in this section. An observable change of state t, or *transition*, is defined by two disjoint sets of places ${}^\bullet t \subseteq P$ and $t^\bullet \subseteq P$. The set ${}^\bullet t$ (the *pre-set* of t) specifies the set of properties which are changed from valid to invalid. The set $t^\bullet$ (the *post-set* of t) specifies the set of properties which are changed from invalid to valid.

By convention, places p are drawn as circles containing a black token when they are marked, and transitions t are drawn as rectangles, with incoming arcs from each place in ${}^\bullet t$ and outgoing arcs towards each place in $t^\bullet$. An elementary net with set of places P and set of transitions T may be defined thus as a graph $(P \cup T, F)$ with a set of arcs $F \subseteq P \times T \cup T \times P$ where ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$ and $t^\bullet = \{p \in P \mid (t,p) \in F\}$. As places and transitions form disjoint sets, such graphs are bi-partite. To stress this, elementary nets may be defined equivalently as triples $N = (P,T,F)$ where P and T are finite disjoint sets (of places and transitions, respectively) and $F \subseteq (P\times T)\cup(T\times P)$ is the set of *flow* arcs. Keeping in mind that transitions represent observable changes of states, two requirements should be stated: every transition t must be connected to at least one place, and it should not be connected to a place by flow arcs in both directions. Moreover, in order to represent unambiguously observable changes of states, different transitions must have different pre-sets, or different post-sets, or both. Altogether, one obtains the following definition.

Definition 1.1. *An* elementary net *is a triple* $N = (P,T,F)$ *where* P *and* T *are finite disjoint sets of* places *and* transitions*, respectively, and* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of* flow arcs*, such that the following requirements are satisfied, letting* ${}^\bullet x = \{y \mid (y,x) \in F\}$ *and* $x^\bullet = \{y \mid (x,y) \in F\}$ *for every* $x \in P \cup T$*:*

1. *there is* **no self-loop***: for every $x \in P \cup T$, ${}^\bullet x \cap x^\bullet = \emptyset$,*
2. *there is* **no isolated transition***: for every $t \in T$, there exists $p \in P$ such that $(p,t) \in F$ or $(t,p) \in F$, i.e. ${}^\bullet t \cup t^\bullet \neq \emptyset$,*
3. *there are* **no equivalent transitions***: for every $t, t' \in T$, if ${}^\bullet t = {}^\bullet t'$ and $t^\bullet = t'^\bullet$, then $t = t'$.*

A marking *of N is any subset of P.* ◇

As elementary nets are bipartite graphs, the condition ${}^\bullet x \cap x^\bullet = \emptyset$ which excludes *self-loops* is indeed satisfied for every element $x \in P \cup T$ as soon as it is satisfied for every transition $x \in T$, or for every place $x \in P$.

Remark 1.2. Since elementary nets were introduced in [33, 34, 31], slightly different definitions have been given in the literature. In [28], condition (2) requiring that no transition should be *isolated* is strengthened by stipulating that no place is isolated either, the condition excluding *self-loops* does not appear, and condition (3) requiring that elementary nets should be *transition simple* is strengthened by demanding that they are also *place simple* (${}^\bullet p = {}^\bullet p'$ and $p^\bullet = p'^\bullet$ entail $p = p'$). In [32], condition (2) is reinforced by stipulating that every transition has a non-empty pre-set *and* a non-empty post-set, and condition (3) does not appear. These variations are more technical than fundamental and they have only marginal influence on the theory of elementary nets and their synthesis. Here, for simplicity, we have chosen the minimal conditions that fit with the interpretation of elementary nets as representations of partial 2-structures [21, 22]. We shall examine this connection in Sec. 1.8. □

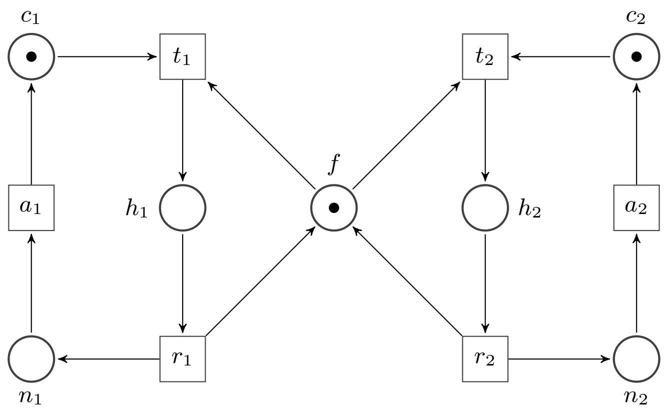


Fig. 1.5. an elementary net system for mutual exclusion

Example 1.3. For an illustration, consider the elementary net shown in Fig. 1.5. This elementary net models two process components with mutually exclusive access to a shared resource. The set of places is $P = \{c_1, h_1, n_1, f, c_2, h_2, n_2\}$ and the set of transitions is $T = \{t_1, r_1, a_1, t_2, r_2, a_2\}$. Place f, when it is marked, indicates that the resource is **free**. Places c_i, h_i, n_i (for $i = 1, 2$) represent the properties of process component i to be **candidate** to the resource, to **hold** the resource, and to have **no need** of the resource, respectively. Transition t_i ($i = 1, 2$) models a state change in which process component i, which was candidate to the resource, **takes** this resource (which was free). Transition r_i (for $i = 1, 2$) models a state change in which process component i, which had the resource, **releases** the resource (which becomes free). Transition a_i (for $i = 1, 2$) models a state change in which process component i **applies** to the resource. All conditions stated in Def. 1.1 are satisfied: there are no isolated transitions, no self-loops, and the net is transition simple. The elementary net shown in Fig. 1.5 is provided with a marking $M_0 \subseteq P$, namely $M_0 = \{c_1, f, c_2\}$, as indicated by the presence of black tokens in the considered places. $\square$

An elementary net with an initial marking is called an elementary net system.

Definition 1.4. *An* elementary net system *is a quadruple $N = (P, T, F, M_0)$ where (P, T, F) is an elementary net, called the* underlying net *of N, and $M_0 \subseteq P$ is the* initial marking*, such that the following requirements are satisfied:*

1. *there are* **no equivalent places***: for every $p, p' \in P$, if ${}^\bullet p = {}^\bullet p'$ and $p^\bullet = p'^\bullet$ and $M_0(p) = M_0(p')$, then $p = p'$,*
2. *there are* **no dead transitions***: see Def. 1.10 given later in this section.*

Two elementary net systems $N = (P, T, F, M_0)$ and $N' = (P', T, F', M'_0)$ with the same set of transitions T are isomorphic, noted $N \cong N'$, if there exists a bijection $\varphi : P \to P'$ such that ${}^\bullet p = {}^\bullet \varphi(p)$, $p^\bullet = \varphi(p)^\bullet$, and $M_0(p) = M'_0(\varphi(p))$ for all $p \in P$. $\diamondsuit$

Transitions of elementary net systems specify *potential* changes of states or markings, but only the initial marking is given explicitly in the definition of the net. The *actual* changes of states may be discovered by applying inductively from the initial marking M_0 the firing rule defined below.

Definition 1.5. *Let $N = (P, T, F, M_0)$ be an elementary net system. A transition $t \in T$ is* enabled *in marking M, notation: $M[t\rangle$, if ${}^\bullet t \subseteq M$ and $M \cap t^\bullet = \emptyset$. A transition t which is enabled in a marking M can be* fired *in M, leading to the marking $M' = (M \setminus {}^\bullet t) \cup t^\bullet$, notation: $M[t\rangle M'$.* $\diamondsuit$

Note that, for any markings M, M' and for any transition t, $M[t\rangle M'$ if and only if $M \setminus M' = {}^\bullet t$ and $M' \setminus M = t^\bullet$. This characterization is a straightforward consequence of Def. 1.5.

The inductive application of the firing rule to an elementary net system produces an initialized transition system, called the reachability graph of the net system.

Definition 1.6. *An* initialized transition system $A = (S, E, \varDelta, s_0)$ *consists of a set of* states S*, a set of* events E*, a set of* transitions $\varDelta \subset S \times E \times S$*, and an initial state* $s_0 \in S$*. The* language *of* A *(notation* $L(A)$*) is the set of all event sequences* $e_1 \dots e_n$ *(including the empty sequence* ε*) such that* $(s_{i-1}, e_i, s_i) \in \varDelta$ *for all* $1 \leq i \leq n$ *for some corresponding sequence of states* $s_1 \dots s_n$*. Two initialized transition systems* $A = (S, E, \varDelta, s_0)$ *and* $A' = (S', E, \varDelta', s_0')$ *with the same set of events* E *are* isomorphic*, noted* $A \cong A'$*, if there exists a bijection* $\varphi : S \rightarrow S'$ *such that* $s_0' = \varphi(s_0)$ *and for all* $s_1, s_2 \in S$ *and* $e \in E$*,* $(s_1, e, s_2) \in \varDelta$ *if and only if* $(\varphi(s_1), e, \varphi(s_2)) \in \varDelta'$*.* ♢

Definition 1.7. *Given an elementary net system* $N = (P, T, F, M_0)$*, the* reachability set *of* N*, noted* $RS(N)$*, is the least set of markings containing the initial marking* M_0 *and closed under the firing of transitions. Markings* $M \in RS(N)$ *are called* reachable *markings of* N*. The* reachability graph *of* N *is the initialized transition system* $RG(N) = (RS(N), T, \varDelta, M_0)$ *defined by letting* $(M, t, M') \in \varDelta$ *if and only if* $M \in RS(N)$ *and* $M[t\rangle M'$*. The* language *of* N*, noted* $L(N)$*, is the language of* $RG(N)$*.* ♢

Definition 1.8. *An* elementary transition system *is an initialized transition system which is isomorphic to the reachability graph of an elementary net system.* ♢

The firing rule allows to generate from an elementary net system an initialized transition system that represents its behaviour extensionally. The (converse) goal of elementary net synthesis is to extract from an initialized transition system an elementary net system which represents this behaviour intensionally.

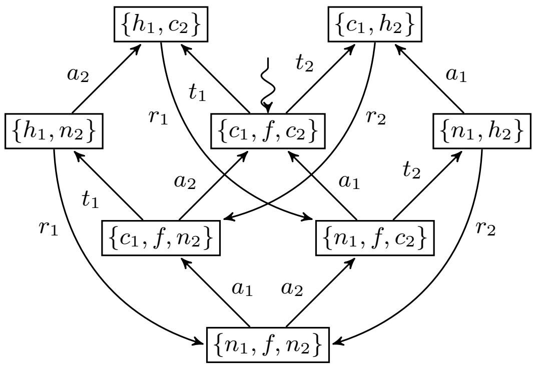


Fig. 1.6. the reachability graph of the elementary net system for mutual exclusion

Example 1.9. For an illustration, we show in Fig. 1.6 the reachability graph of the elementary net system for mutual exclusion (see Fig. 1.5). The initial marking is indicated with a wriggling arrow. It may be checked from the reachability graph that a mutually exclusive use of the resource by the two process components is actually enforced, because the two properties represented by the respective places h_1 (process component 1 has the resource) and h_2 (process component 2 has the resource) never hold jointly. Note that every transition of the elementary net system for mutual exclusion is initially live, i.e., it can be fired in some reachable marking. Even better, every transition is live, i.e. it stays initially live whatever reachable marking is chosen to replace the initial marking. □

Definition 1.10. *Given a net system $N = (P, T, F, M_0)$, a transition $t \in T$ is* dead *if it cannot be fired in any reachable marking $M \in RS(N)$. A net system without dead transitions is said to be* initially live *or* reduced. ♢

By Def. 1.4, elementary net systems are free from dead transitions. The other particularities of reachability graphs of elementary net systems will be examined in Sect. 1.3, where we will refine consequently the current definition of initialized transition systems.

In the end of the section, we would like to discuss the issue of *contacts*, which is special to elementary nets. In an elementary net system, a transition t may be disabled in a reachable marking M because some input place of t is not marked in M (${}^\bullet t \not\subseteq M$) or because some output place of t is marked in M ($t^\bullet \cap M \neq \emptyset$). The latter case is known as contact.

Definition 1.11. *Given an elementary net system $N = (P, T, F, M_0)$ and a reachable marking M of N, a transition $t \in T$ is said to have* contact *in M if ${}^\bullet t \subseteq M$ and $t^\bullet \cap M \neq \emptyset$. N is said to be* contact-free *if ${}^\bullet t \subseteq M$ entails $t^\bullet \cap M = \emptyset$ for every transition $t \in T$ and for every reachable marking M.* ♢

Example 1.12. Figure 1.7, borrowed from [19], shows two elementary net systems with isomorphic reachability graphs (see Def. 1.6). The elementary net system on the left is contact-free. The elementary net system on the right has contacts. Indeed, transition *summer* has no input place, but it cannot be fired because one of its output places already contains a token. After transition *spring* has been fired, this token is removed and transition *summer* can be fired. □

Definition 1.13. *Given elementary net systems $N = (P, T, F, M_0)$ and $N' = (P', T', F', M_0')$* with the same set of transitions *$T = T'$, let $N \approx N'$ if and only if $RG(N) \cong RG(N')$, i.e., N and N' have isomorphic reachability graphs.* ♢

Up to the equivalence relation $\approx$, contacts may always be eliminated from elementary net systems by introducing *complementary places* defined as follows.

Definition 1.14. *In an elementary net system $N = (P, T, F, M_0)$, two places p and p' are said to be* complementary *if $p \in M_0 \Leftrightarrow p' \notin M_0$ and ${}^\bullet p = p'^\bullet$ and $p^\bullet = {}^\bullet p'$.* ♢

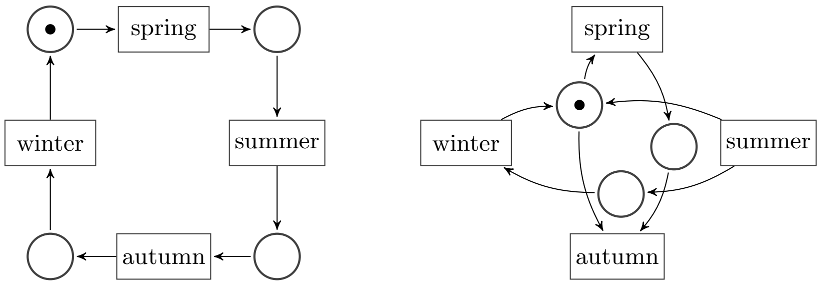


Fig. 1.7. two alternative elementary net systems for the *four seasons*

It may be shown by induction on the length of firing sequences that, if p and p' are complementary places, then $p \in M \Leftrightarrow p' \notin M$ for every reachable marking M. Therefore, if M is a reachable marking of N and if every place p of N has a complementary place p', then whenever $(t, p) \in F$ and $p \in M$, $(p', t) \in F$ and $p' \notin M$, entailing that ${}^\bullet t \nsubseteq M$ and that transition t has no contact in M.

Given an elementary net system $N = (P, T, F, M_0)$, let N' denote the elementary net system (extending N) which one obtains by adding, for each place p of N, the complementary place p' if it is not already present in P. Then N' is contact-free. For each marking $M \in RS(N)$, let M' denote the unique marking of N' such that, for every place $p \in P$, $p \in M'$ iff $p \in M$, and for every (complementary) place $p' \notin P$, $p' \in M'$ iff $p \notin M$. Then $M' \in RS(N')$ and the correspondence $M \mapsto M'$ defines an isomorphism between the reachability graphs $RG(N)$ and $RG(N')$. Therefore, $N \approx N'$ as desired.

Example 1.15. The contact-free elementary net system which is obtained by adding complementary places to the elementary net system displayed on the right-hand side of Fig.1.7 is shown in Fig. 1.8. Transition *summer* cannot be fired because one of its input places is not marked. □

Isomorphism of reachability graphs is a strong equivalence on elementary net systems. A weaker equivalence on elementary net systems, also used intensively, is *language equivalence*.

Definition 1.16. *Two elementary net systems N and N'* with the same set of transitions $T = T'$ *are* language equivalent *(notation: $N \sim N'$) if $L(N) = L(N')$.* ♢

Clearly, $N \approx N'$ entails $N \sim N'$, but not the other way round. The two elementary net systems shown in Fig. 1.9 have identical languages but non-isomorphic reachability graphs (see Fig. 1.10). The reachability graph of the rightmost net system is the minimal automaton recognizing the language $\{\varepsilon, a, b, ac, bd\}$.

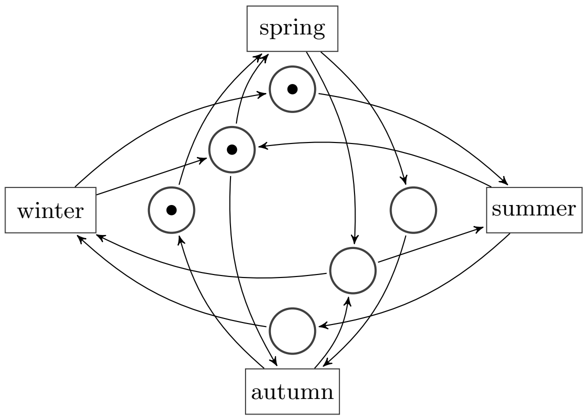


Fig. 1.8. the complemented version of the net system on the right of Fig. 1.7

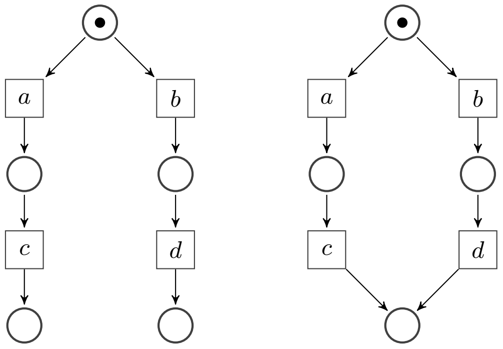


Fig. 1.9. two alternative net systems for the language $\{\varepsilon, a, b, ac, bd\}$

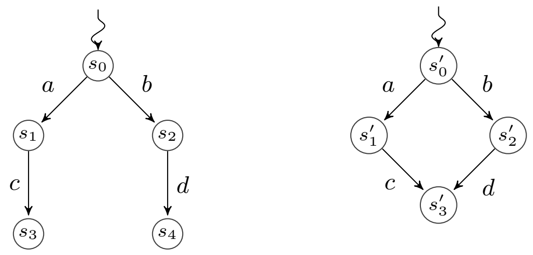


Fig. 1.10. the reachability graphs of the two nets from Fig. 1.9

1.3 Regions and Elementary Transition Systems

After two preliminary sections, let us now introduce the main topic of this book. The *basic synthesis problem* for elementary net systems consists of deciding whether a given finite initialized transition system is isomorphic to the reachability graph of some elementary net system. In this case, the transition system is called an *elementary transition system* (Def. 1.8) and the net system is said to be a *net realization* of the transition system. This problem is decidable since, for a given initialized transition system $A = (S, E, \varDelta, s_0)$, there can exist only finitely many elementary net systems $N = (P, T, F, M_0)$ with set of transitions T equal to the set of events E. However, enumerating these elementary net systems N, computing their reachability graphs $RG(N)$, and checking whether A is isomorphic to some of them, is not a smart way to solve the problem. Another possibility is to derive from A a unique and optimal candidate $N = SN(A)$ for the realization of A (SN means Synthesized Net) and to check whether A and $RG(SN(A))$ are isomorphic. We show in this section that this can actually be done using *regions* of transition systems as defined in [21, 22].

The best way to explain intuitions under the notion of regions is to consider the basic synthesis problem (for elementary net systems) in the special case where the initialized transition system $A = (S, E, \varDelta, s_0)$ taken as input **is** the reachability graph of an elementary net system $N = (P, T, F, M_0)$, i.e., $T = E$ and $RG(N) = A$. The idea is to examine, for each place $p \in P$, the set $[\![p]\!]$ of markings of N in which place p is marked, to look at the graph theoretical properties of $[\![p]\!]$ within $RG(N)$, and to reconstruct on the sole basis of these graph theoretical properties the initial marking and the flow arcs attached to place p.

Definition 1.17. *Let $N = (P, T, F, M_0)$ be an elementary net system. For any place $p \in P$, the set $[\![p]\!] = \{M \in RS(N) \mid p \in M\}$ is called the* extension *of p in $RG(N)$.* ♢

Example 1.18. Consider again the elementary net system for mutual exclusion (right hand side of Fig. 1.11). Place f of this net is marked when the resource is **free**. The extension $[\![f]\!]$ of place f is figured by the grey slots in the reachability graph of the net (left hand side of Fig. 1.11). One observes that all occurrences of the events r_1 or r_2 (**release** the resource) *enter* the extension of place f, thus witnessing that $r_1, r_2 \in {}^\bullet f$. Similarly, all occurrences of the events t_1 or t_2 (**take** the resource) *exit from* the extension of place f, thus witnessing that $t_1, t_2 \in f^\bullet$. The occurrences of the remaining events a_1 and a_2 (**apply to** the resource) *do not cross the border of* $[\![f]\!]$, thus witnessing that $a_1, a_2 \notin {}^\bullet f \cup f^\bullet$. □

By Def. 1.17, every place p defines a property which is shared by all markings in its extension and by no other reachable markings. By Def. 1.4, two different places of an elementary net system must have different extensions. Places p are

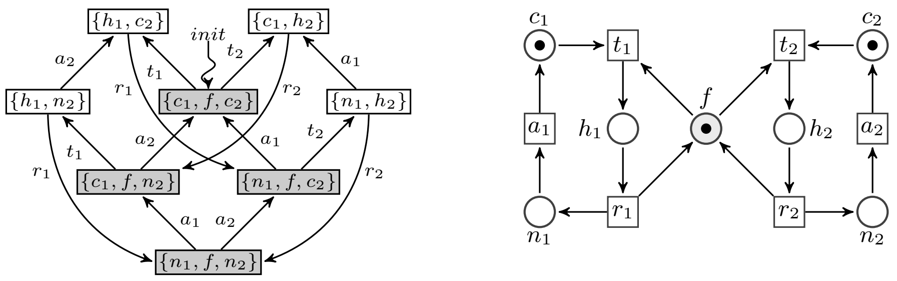


Fig. 1.11. the extension of a place in a reachability graph

therefore faithfully represented by their extensions $[\![p]\!]$, i.e., by sets of states of $RG(N)$. The next proposition, generalizing over the observations made in Example 1.18, establishes much stronger relations between the places of an elementary net system N and their extensions in $RG(N)$.

Proposition 1.19. *Let $N = (P, E, F, M_0)$ be an elementary net system, and let $RG(N) = (S, E, \Delta, M_0)$ be the reachability graph of N (hence $S = RS(N)$). For every place $p \in P$ and for every $e \in E$ (transition of N and event of $RG(N)$), the following relations are satisfied:*

$$\begin{aligned}
e \in {}^\bullet p &\quad\Leftrightarrow\quad \big(s \xrightarrow{e} s' \Rightarrow (s \notin [\![p]\!] \wedge s' \in [\![p]\!])\big)\\
e \in p^\bullet &\quad\Leftrightarrow\quad \big(s \xrightarrow{e} s' \Rightarrow (s \in [\![p]\!] \wedge s' \notin [\![p]\!])\big)\\
e \notin {}^\bullet p \cup p^\bullet &\quad\Leftrightarrow\quad \big(s \xrightarrow{e} s' \Rightarrow (s \in [\![p]\!] \Leftrightarrow s' \in [\![p]\!])\big)
\end{aligned}$$

Proof. According to the firing rule of elementary net systems,

$$M[e\rangle M' \Leftrightarrow (M \setminus M' = {}^\bullet e \text{ and } M' \setminus M = e^\bullet)$$

Therefore,

$$\begin{aligned}
(M[e\rangle M' \ \wedge\ p \in e^\bullet) &\quad\Rightarrow\quad (p \notin M \ \wedge p \in M')\\
(M[e\rangle M' \ \wedge\ p \in {}^\bullet e) &\quad\Rightarrow\quad (p \in M \ \wedge p \notin M')\\
(M[e\rangle M' \ \wedge\ p \notin {}^\bullet e \cup e^\bullet) &\quad\Rightarrow\quad (p \in M \ \Leftrightarrow p \in M')
\end{aligned}$$

which may be reformulated as

$$\begin{aligned}
p \in e^\bullet &\quad\Rightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \notin M \ \wedge p \in M'))\\
p \in {}^\bullet e &\quad\Rightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \in M \ \wedge p \notin M'))\\
p \notin {}^\bullet e \cup e^\bullet &\quad\Rightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \in M \ \Leftrightarrow p \in M'))
\end{aligned}$$

As the conditions on the left-hand side are mutually exclusive and cover all possible cases for each place p, and similarly for the conditions in the right-hand side, the implication relations may be replaced by logical equivalence relations:

$$\begin{aligned}
p \in e^\bullet &\quad\Leftrightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \notin M \ \wedge p \in M'))\\
p \in {}^\bullet e &\quad\Leftrightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \in M \ \wedge p \notin M'))\\
p \notin {}^\bullet e \cup e^\bullet &\quad\Leftrightarrow\quad (M[e\rangle M' \ \Rightarrow\ (p \in M \ \Leftrightarrow p \in M'))
\end{aligned}$$

This is a close rephrasing of the statement of the proposition, which may be derived using the relations $p \in {}^\bullet e \Leftrightarrow e \in p^\bullet$, $p \in M \Leftrightarrow M \in [\![p]\!]$, and so on. □

Proposition 1.19 states that the flow relations of an elementary net system N may be reconstructed from its reachability graph. It suffices in fact, for each place p, to classify the transitions e of N according to whether all occurrences of the corresponding event e enter jointly, or leave jointly, or do not cross the border of $[\![p]\!]$, the extension of p in $RG(N)$. Note that a place p contains a token in the initial marking M_0 of N if and only if the initial state M_0 of $RG(N)$ belongs to $[\![p]\!]$. Therefore, N may be entirely reconstructed from $RG(N)$.

Consider now an initialized transition system $A = (S, E, \Delta, s_0)$ and suppose that $A \cong RG(N)$ for some elementary net system $N = (P, E, F, M_0)$. Let $\varphi : S \to RS(N)$ be the isomorphism between A and $RG(N)$. In view of Prop. 1.19, for every place $p \in P$, the set of states $\varphi^{-1}[\![p]\!]$ is a region of A according to the following definition, due to Ehrenfeucht and Rozenberg [21, 22].

Definition 1.20. *A region of an initialized transition system* (S, E, Δ, s_0) *is a subset of states* $r \subseteq S$ *such that, for each event* $e \in E$*, one of the following mutually exclusive situations holds:*

e **enters region** r *, noted* $e \in {}^\circ r$*, which means that*

$$s \xrightarrow{e} s' \quad \Rightarrow \quad (s \notin r \ \wedge \ s' \in r)$$

e **exits from region** r *, noted* $e \in r^\circ$*, which means that*

$$s \xrightarrow{e} s' \quad \Rightarrow \quad (s \in r \ \wedge \ s' \notin r)$$

e **does not cross the border of region** r *, noted* $e \in r^\perp$*, which means that*

$$s \xrightarrow{e} s' \quad \Rightarrow \quad (s \in r \ \Leftrightarrow \ s' \in r)$$

The whole set of states S *and its complement, the empty set, are the* trivial *regions. The set of regions of an initialized transition system* A *is denoted* $R(A)$*. For any event* e *of* A*, the subsets of regions* ${}^\circ e = \{r \in R(A) \mid e \in r^\circ\}$ *and* $e^\circ = \{r \in R(A) \mid e \in {}^\circ r\}$ *are called the* preset*, respectively the* poset*, of* e*. Similarly, for any region* r *of* A*, the subsets of events* ${}^\circ r$ *and* r° *are called the* preset*, respectively the* poset *of* r*. Finally, for any subset of regions* $R \subseteq R(A)$ *and for every state* $s \in S$*, we let* $R_s = \{r \in R \mid s \in r\}$*.* ♢

Note that Def. 1.20 applies to *arbitrary* initialized transition systems, hence also to non-elementary transition systems, i.e., it applies also to initialized transition systems that do not have net realizations. A very basic illustration of Def. 1.20 is given in Fig. 1.12, where the outer rectangle represents the set

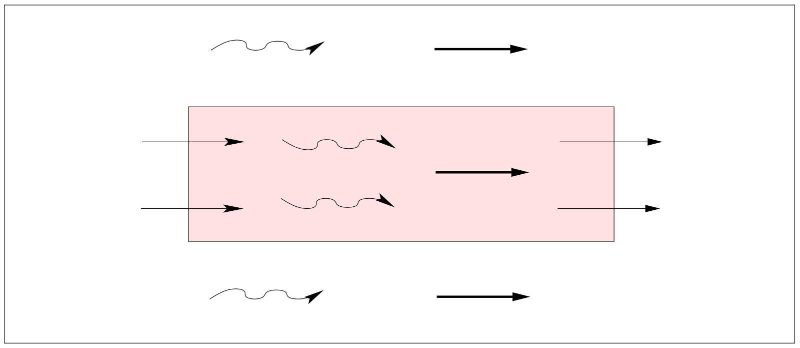


Fig. 1.12. events crossing or not crossing the borders of a region

of states of an initialized transition system, while the inner and grey rectangle represents a region. Each type of arcs in the figure represents the collection of all transitions (s, e, s') labelled with the same event e. All possible cases for such a collection are shown in the figure. It is easily seen from Def. 1.20 and Fig. 1.12 that for any region $p \in R(A)$, the complement $\overline{p} = S \setminus p$ of p is also a region of A. Moreover, ${}^\circ\overline{p} = p^\circ$ and $\overline{p}^\circ = {}^\circ p$.

Example 1.21. Consider the initialized transition system shown in Fig. 1.13. Each one of subsets of states $\{s_0\}$, $\{s_1, s_3\}$, $\{s_2, s_4\}$ is a region. On the con-

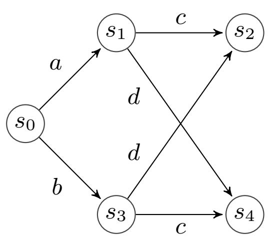


Fig. 1.13. a non-elementary transition system

trary, $\{s_1, s_2\}$ is not a region: one d-transition enters the set but the other d-transition leaves the set. The set $\{s_1, s_2, s_4\}$ is not a region either, since one d-transition enters this set but the other d-transition does not cross the border of the set. The set $\{s_1, s_2, s_3, s_4\}$ is a region, and so are the sets $\{s_0, s_1, s_3\}$ and $\{s_0, s_2, s_4\}$. There are no other non-trivial regions. □

The notion of regions is graph-theoretic, i.e., regions are preserved under isomorphisms of graphs. So, the definition of the regions of a transition system does not depend on the inscriptions possibly attached to its states. On the contrary, the definition of the extension of a place in a reachability graph relies entirely on the fact that states are markings. Nevertheless, for any place

p of an elementary net system N, the extension $[\![p]\!]$ of p in $RG(N)$ is a region of $RG(N)$. By Prop. 1.19, $e \in {}^\bullet p \Leftrightarrow e \in {}^\circ[\![p]\!]$, $e \in p^\bullet \Leftrightarrow e \in [\![p]\!]^\circ$, and $e \notin {}^\bullet p \cup p^\bullet \Leftrightarrow e \in [\![p]\!]^\perp$. Moreover, $p \in M_0 \Leftrightarrow s_0 \in [\![p]\!]$. This gives precise rules for reconstructing an elementary net system N from regions $[\![p]\!]$ of $RG(N)$ induced from places p of N. However, if one considers all regions of $RG(N)$, and not only the regions that coincide with extensions of places of N, then by applying the considered reconstruction or resynthesis rules, one generally obtains a net larger than the original net N.

Example 1.22. Let $A = RG(N)$ be the reachability graph of the elementary net system N for mutual exclusion (see Fig. 1.11). The regions of A are the extensions of the places of N, the complements of the latter, and the two trivial regions. The net system resynthesized from all regions of A, denoted $SN(A)$, is depicted in Fig. 1.14. In this figure, X is the place of $SN(A)$ defined from the region $[\![x]\!]$ of A, for every place x of N, and $\overline{X}$ is the complementary place. The original net system N coincides with the restriction of the resynthesized net system $SN(RG(N))$ on the subset of places defined from extensions of places of N. □

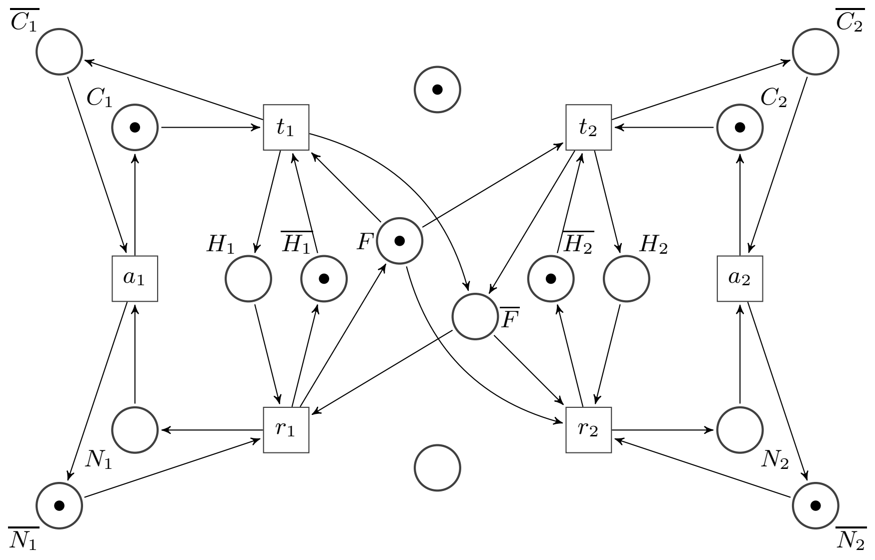


Fig. 1.14. the net system resynthesized from the reachability graph of the net system for mutual exclusion (Fig. 1.5)

We now apply the rules used to reconstruct N from $RG(N)$ to an arbitrary initialized transition system A, with the objective of synthesizing from the regions A an elementary net system $SN(A)$ realizing A, or realizing A as closely as possible. Note that A may not be an elementary transition system, i.e., it may not be isomorphic to the reachability graph of any elementary net system. A difficulty arises here because, if one applies the synthesis rules like was done in Example 1.21 to a non-elementary initialized transition system A, then the result is generally not an elementary net system that conforms to Defs. 1.1 and 1.4. For this reason, we must propose extended definitions.

Definition 1.23. *A* quasi-elementary net *is a triple $N = (P, T, F)$ like in Def. 1.1, except that isolated or equivalent transitions may be present. A* quasi-elementary net system *is a net system $N = (P, T, F, M_0)$ like in Def. 1.4, except that the underlying net (P, T, F) is only required to be quasi-elementary. The net firing rule and all notations given for elementary nets and elementary net systems are extended without any change to quasi-elementary nets and to quasi-elementary net systems.* $\diamondsuit$

In Part I of this book, from now on, *net* and *net system* mean by default *quasi-elementary net* and *quasi-elementary net system*. Thus, '*elementary*' is never meant unless stated explicitly.

Definition 1.24. *An initialized transition system is a* quasi-elementary transition system *if it is isomorphic to the reachability graph of a quasi-elementary net system.* $\diamondsuit$

With the above definitions, we can now define synthesized nets.

Definition 1.25. *The* canonical net version $SN(A) = (P, T, F, M_0)$ *of an initialized transition system $A = (S, E, \Delta, s_0)$, also called the net system synthesized from A, is the quasi-elementary net-system defined by $P = R(A)$, $T = E$, $F = \{(p, e) \mid e \in p^\circ\} \cup \{(e, p) \mid e \in {}^\circ p\}$, and $M_0 = \{p \in R(A) \mid s_0 \in p\}$.* $\diamondsuit$

Remark 1.26. For fluidity of the presentation, we postpone the task of showing that Def. 1.25 is consistent with Def. 1.23, which requires indirectly by Def. 1.4 that $SN(A)$ should be free from equivalent places and dead transitions. These properties will be established in Lemma 1.32 and Lemma 1.34, respectively. $\square$

By Def. 1.25, the places of $SN(A)$ are the regions of A, and the transitions of $SN(A)$ are the events of A. Recall that the complement $\overline{p} = S \setminus p$ of a region p is also a region, with ${}^\circ\overline{p} = p^\circ$ and $\overline{p}^\circ = {}^\circ p$. Therefore, in the net system $SN(A)$, ${}^\bullet\overline{p} = p^\bullet$ and $\overline{p}^\bullet = {}^\bullet p$. So, every place p has a complementary place $\overline{p}$ in $SN(A)$, and $SN(A)$ is contact-free. The trivial regions of A, seen as places of $SN(A)$, are isolated (because ${}^\circ S \cup S^\circ = {}^\circ\emptyset \cup \emptyset^\circ = \emptyset$). The trivial place S is initially marked and keeps marked in all reachable markings. The trivial place $\emptyset$ is initially unmarked and keeps unmarked in all reachable markings.

Example 1.27. Consider the initialized transition system A shown on the left of Fig. 1.15. The canonical net version $SN(A)$ of A is shown in the middle of the figure. This net system is not elementary, since it has one isolated transition (c) and two equivalent transitions (a and b). The reachability graph $RG(SN(A))$ is shown on the right of the figure. □

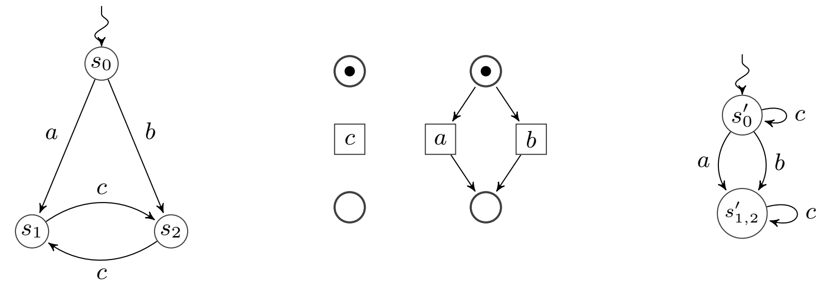


Fig. 1.15. A, $SN(A)$, and $RG(SN(A))$

Example 1.28 (Exple. 1.21 continued). The quasi-elementary net system shown on the left of Fig. 1.16 is the canonical net version of the initialized transition system A shown in Fig. 1.13. The reachability graph of this net is depicted on the right of Fig. 1.16, showing (in view of the arguments presented below in this section) that A is not an elementary transition system. □

We will prove that for any initialized transition system A, the synthesized net system $SN(A)$ is an optimal candidate for the exact realization of A, both in the class of elementary net systems or in the class of quasi-elementary net systems. In other words, if A is isomorphic to $RG(N)$ for some net system N, then $A \cong RG(SN(A))$. By Def. 1.24, A is a quasi-elementary transition system in this case. Moreover, if A is an elementary transition system, i.e., if $A \cong RG(N)$ for some elementary net system N, then $SN(A)$ is an elementary net system. We establish in reverse order the second claim (Prop. 1.31) and the first claim (Th. 1.37). Before this, we need introducing more definitions.

In view of the net firing rule, an initialized transition system which may be realized by a quasi-elementary net system must be *deterministic*. Moreover, an initialized transition system which may be realized by an elementary net system must be *loop-free* and *simple*. Let us give precise meanings to these terms.

Definition 1.29. *An initialized transition system* (S, E, Δ, s_0) *is said to be* deterministic *if, for every event* e, $s \xrightarrow{e} s' \wedge s \xrightarrow{e} s'' \Rightarrow s' = s''$ *for all states* s, s', s" *where* $s \xrightarrow{e} s'$ *means* $(s, e, s') \in \Delta$. *We let* $\delta : S \times E \to S$ *denote the partial function such that* $\delta(s, e) = s'$ *if* $(s, e, s') \in \Delta$ *for some* s' *(thus unique)*

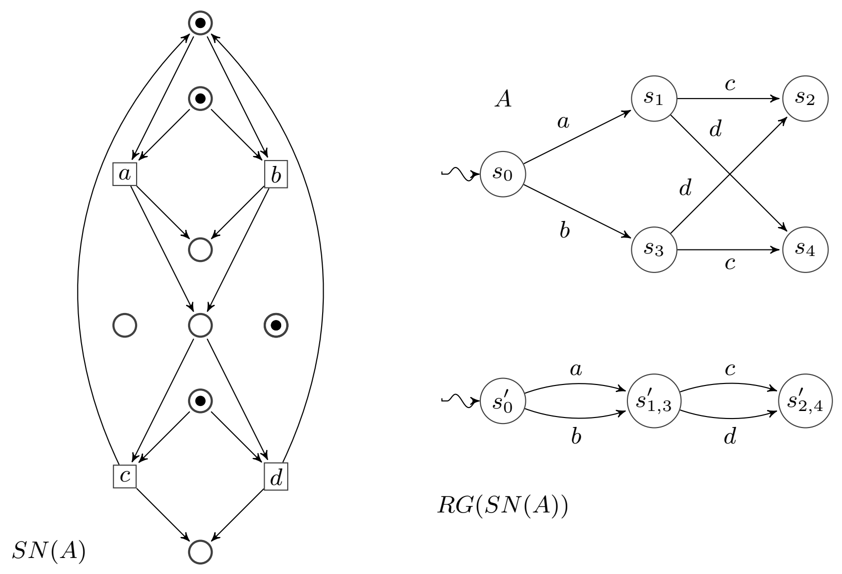


Fig. 1.16. a quasi-elementary net system and its reachability graph

and $\delta(s,e)$ is undefined otherwise. Notations (S,E,Δ,s_0) and $(S,E,\delta' s_0)$ are used indifferently. An event e is enabled *in state s (notation: $s \xrightarrow{e}$) if $\delta(s,e)$ is defined,* disabled *otherwise. The map δ is extended inductively to sequences of events by letting $s \xrightarrow{\varepsilon} s$ (where ε denotes the empty sequence) and $s \xrightarrow{e \cdot u} s'$ for all $e \in E$ and $u \in E^*$ such that $s \xrightarrow{e}$ and $\delta(s,e) \xrightarrow{u} s'$. For $s \in S$, the set of sequences $u \in E^*$ for which $\delta(s,u)$ is defined is denoted $L(s)$. State s' is* reachable *from state s if $s \xrightarrow{u} s'$ for some (possibly empty) sequence $u \in L(s)$.* $\diamondsuit$

In the rest of this book, initialized transition systems are always assumed to be *deterministic*

Definition 1.30. *Let $A = (S,E,\delta,s_0)$ be an initialized transition system. A is said to be* reachable *if all states in S are reachable (from s_0). If A is reachable, then it is said to be reduced if, for every event $e \in E$, $\delta(s,e)$ is defined for some $s \in S$. A is* loop free *if, for every event e, $s \xrightarrow{e} s' \Rightarrow s \neq s'$. A is* simple *if, for every pair of distinct events e and e', $s \xrightarrow{e} s' \wedge s \xrightarrow{e'} s' \Rightarrow e = e'$.* $\diamondsuit$

In the rest of this book, initialized transition systems are always assumed to be *reachable* and *reduced*

The next proposition states two structural properties of elementary transition systems that may serve as criteria for rejecting immediately many initialized transition systems that cannot be realized exactly by elementary net systems.

Proposition 1.31. *For any elementary net system N, the reachability graph $RG(N)$ of N is a loop-free and simple initialized transition system. Conversely, if A is a loop-free and simple initialized transition system, then any net realization N of A is an elementary net system.*

Proof. As we consider exclusively initially live net systems (see Def. 1.10), for any net system N, the reachability graph $RG(N)$ of N is reachable and reduced. We show that, if N is an elementary net system , i.e., N has neither isolated nor equivalent transitions, then $RG(N)$ is loop-free and simple. If $M[e\rangle M$ for some marking $M \in RS(N)$, then ${}^\bullet e = M \setminus M = \emptyset$ and similarly $e^\bullet = M \setminus M = \emptyset$, hence e is an isolated transition. If $M[e_1\rangle M'$ and $M[e_2\rangle M'$ for some marking $M \in RS(N)$, then ${}^\bullet e_1 = M \setminus M' = {}^\bullet e_2$ and $e_1{}^\bullet = M' \setminus M = e_2{}^\bullet$, hence $e_1 = e_2$. Conversely, let A be a loop-free and simple initialized transition system, and let N be a net system realization of A. We show that N has neither isolated nor equivalent transitions. Assume for contradiction that ${}^\bullet e = e^\bullet = \emptyset$ for some transition e of N. As A is an initialized transition system, by the assumptions made in Def. 1.30, the event e is enabled in some reachable state of A. As $A \cong RG(N)$, $M[e\rangle M'$ for some marking $M \in RS(N)$, and necessarily $M = M'$ since ${}^\bullet e = e^\bullet = \emptyset$. Therefore $M[e\rangle M$. As $A \cong RG(N)$ and A is loop-free, we have reached a contradiction. Assume for contradiction that ${}^\bullet e_1 = {}^\bullet e_2$ and $e_1{}^\bullet = e_2{}^\bullet$ for $e_1 \neq e_2$. Let $M \in RS(N)$ such that $M[e_1\rangle M'$, then necessarily, $M[e_2\rangle M'$. As $A \cong RG(N)$, necessarily $s \xrightarrow{e_1} s'$ and $s \xrightarrow{e_2} s'$ in A for two states s and s' corresponding with M and M', respectively. As A is simple, we have reached a contradiction. □

In view of Prop. 1.31, solving the elementary net synthesis problem for an initialized transition system A can be done in two steps. In a first step, one checks whether A is loop-free and simple (these conditions are necessary for the existence of elementary net realizations of A). In a second step, one searches for net realizations of A without worrying about isolated or equivalent transitions (which are proscribed in elementary net systems). Indeed, if A is loop-free and simple, then any net system realization N of A is an elementary net system.

We focus henceforth our attention on the quasi-elementary net synthesis problem (as opposed to the elementary net synthesis problem) for initialized transition systems. We show first that Def. 1.25 is indeed consistent with Def. 1.23, i.e., that for any initialized transition system A conforming to Def. 1.30, the synthesized net $SN(A)$ is free from equivalent places and dead transitions.

Lemma 1.32. *Two non-trivial regions of an initialized transition system with the same preset and the same postset are equal.*

Proof. Let $A = (S, E, \delta, s_0)$ be an initialized transition system. Let r and r' be two non-trivial regions of A with identical presets and identical postsets. We show first that $s_0 \in r$ if and only if $s_0 \in r'$. Suppose for the sake of contradiction that, e.g., $s_0 \in r$ and $s_0 \notin r'$. As r is a non-trivial region, there exists some sequence of events $e_1 \dots e_n$ such that $s_0 \xrightarrow{e_1 \dots e_n} s_n$ for some state $s_n \notin r$. Let $e_1 \dots e_n$ be chosen minimal among such sequences, and let $s_0 \xrightarrow{e_1 \dots e_i} s_i$ for all $i < n$. By induction on i, and by direct application of Def. 1.20 at each step in the induction, $e_i \in r^{\perp}$ and $s_i \in r$ for all $i < n$. Similarly, $e_i \in r'^{\perp}$ (because r and r' have identical presets and identical postsets) and $s_i \notin r'$ for all $i < n$. Now $s_{n-1} \xrightarrow{e_n} s_n$, $s_{n-1} \in r$, and $s_n \notin r$ entail $e_n \in r^{\circ}$. As r and r' have identical postsets, $e_n \in r'^{\circ}$, but $s_{n-1} \xrightarrow{e_n} s_n$ and $s_{n-1} \notin r'$ entail $e_n \notin r'^{\circ}$, hence we have reached a contradiction. So, $s_0 \in r$ if and only if $s'_0 \in r$. Consider any other state $s \in S$. Then $s_0 \xrightarrow{e_1 \dots e_n} s$ for some sequence of events $e_1 \dots e_n$. By induction on n, and by direct application of Def. 1.20 at each step in the induction, $s \in r$ if and only if $s \in r'$. As s was chosen arbitrarily, $r = r'$. □

Lemma 1.33. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system. If $s \xrightarrow{e} s'$ in A, then $R_s[e\rangle R_{s'}$ in $SN(A)$.*

Proof. Recall from Def. 1.20 that $R_s = \{p \in R(A) \mid s \in p\}$. Assume that $s \xrightarrow{e} s'$ in A. From ${}^{\circ}e = \{p \in R(A) \mid s \in p \ \wedge \ s' \notin p\} = R_s \setminus R_{s'}$, and $e^{\circ} = \{p \in R(A) \mid s \notin p \ \wedge \ s' \in p\} = R_{s'} \setminus R_s$ it follows that $R_s[e\rangle R_{s'}$ in $SN(A)$. □

Lemma 1.34. *If A is an initialized transition system, then $SN(A)$ has no dead transitions.*

Proof. Let $A = (S, E, \delta, s_0)$, then $SN(A) = (R(A), E, F, M_0)$ where $M_0 = R_{s_0}$. As A is an initialized transition system that conforms to Def. 1.30, for every $e \in E$, $s_0 \xrightarrow{e_1 \dots e_n} s$ and $s \xrightarrow{e} s'$ in A for some sequence of events $e_1 \dots e_n$. By Lemma 1.33 and by induction on n, it follows that every transition e of $SN(A)$ can be fired in some reachable marking. □

We need introducing two more definitions before we can establish the main result of the section, namely that $SN(A)$ is an optimal candidate for the realization of A by a net system.

Definition 1.35. *Let $N = (P, T, F, M_0)$ be a net system and let $P' \subseteq P$ be a subset of places of N. The* restriction *of N on P' is the net system $N_{/P'} = (P', T, F', M'_0)$ where $F' = F \cap (P' \times T \cup T \times P')$ and $M'_0 = M_0 \cap P'$.* ♢

Definition 1.36. *For any initialized transition system A and for any subset of regions $R \subseteq R(A)$, let $SN_R(A)$ denote the restriction of the synthesized net system $SN(A)$ on places $R \subseteq R(A)$. $SN_R(A)$ is called the net system synthesized from R (and A).* ♢

Theorem 1.37. *The canonical net version $SN(A) = (P, E, F, M_0)$ of an initialized transition system $A = (S, E, \delta, s_0)$, given in Def. 1.25, is the largest candidate for the exact realization of A in the class of net systems:*

1. *any net system $N = (P', E, F', M'_0)$ realizing A ($A \cong RG(N)$) is isomorphic to some restriction of $SN(A)$; more precisely, $N \cong SN_{R_N}(A)$ where R_N is the set of regions of A that correspond with extensions of places of N through the isomorphism between A and $RG(N)$,*
2. *whenever A has some net system realization, $SN(A)$ is a realization of A.*

Proof. By Proposition 1.19, in view of Def. 1.17, any net system $N = (P', E, F', M'_0)$ realizing A is isomorphic to the restriction of $SN(A)$ on $R_N = \{[\![p]\!] \in R(A) \mid p \in P'\}$, hence (1) is proved. In order to establish (2), consider any net system realizing A. By (1), this net system is isomorphic to a restriction of $SN(A)$, let $N = (P', E, F', M'_0)$ where $P' = R_N$. In view of this isomorphism, $A \cong RG(N)$. We want to show that $A \cong RG(SN(A))$. As N is a restriction of $SN(A)$, $M'_0 = R_{s_0} \cap R_N$. As $A \cong RG(N)$ and N is a restriction of $SN(A)$, by inductive application of Lemma 1.33, any reachable marking of N is of the form $R_s \cap R_N$ for some state $s \in S$. Moreover, for the same reasons, $s \xrightarrow{e} s'$ in A if and only if $(R_s \cap R_N)[e\rangle(R_{s'} \cap R_N)$ in N (for all states $s, s' \in S$). In order to prove that $A \cong RG(SN(A))$, we finally show that $s \xrightarrow{e} s'$ in A if and only if $R_s[e\rangle R_{s'}$ in $SN(A)$ (for all states $s, s' \in S$). If $R_s[e\rangle R_{s'}$ in $SN(A)$, then clearly $(R_s \cap R_N)[e\rangle(R_{s'} \cap R_N)$ in N, hence $s \xrightarrow{e} s'$ in A. If $s \xrightarrow{e} s'$ in A, then by Lemma 1.33, $R_s[e\rangle R_{s'}$ in $SN(A)$, hence the proof is complete. □

For concluding the section, we indicate a procedure based on Prop. 1.31 and Th. 1.37 to deal with the elementary net synthesis problem from a finite initialized transition system A. After checking that the initialized transition system A is loop-free and simple, which are two necessary conditions of realizability, one searches for arbitrary net system realizations N of A, knowing by Prop. 1.31 that they, in fact, are always elementary net realizations of A. Such realizations, necessarily finite, exist if and only if the canonical net version of A synthesized from all regions of A is a realization. Therefore, one computes the set $R(A)$ of all regions of A, and one constructs the net system $SN(A)$ synthesized from $R(A)$. The initialized transition system A may be realized by an elementary net system if and only if $A \cong RG(SN(A))$. Since A is deterministic, this isomorphism can be checked on the fly within time linear in the size of A and in the number of places of $SN(A)$, i.e. in $|R(A)|$ which may unfortunately be exponential in the size of A. We prove in Chapter 2 that the problem of elementary net synthesis is indeed NP-complete. Nevertheless, we shall see in the next section that, when some net realization exists, the number of places of this net may be bounded by $|S| \times ((|S| - 1)/2 + |E|)$ where $A = (S, E, \Delta, s_0)$.

1.4 Admissible Sets of Regions and the Separation Axioms

In sections 1.2 and 1.3, we have established a two way connection between initialized transition systems and (quasi-elementary) net systems, comprised of two operators. The first operator $RG(N)$ builds the reachability graph of a net system N, thus associating behaviour to structure. The second operator $SN(A)$ distills an initialized transition system A and produces a net system, thus extracting structure from behaviour. In general, the reachability graph of $SN(A)$ is not isomorphic to A (however, it is as close as possible to A, as will be shown in Sec. 1.5). So, not every initialized transition system may be represented by a net system, and to make the representation useful, it is mandatory to identify those initialized transition systems A which may be given exact net representations, i.e., such that $A \cong RG(SN(A))$. Needless to say, the required characterization should be independent of the operators SN and RG. In this section, we present to this effect an axiomatic characterization of elementary transition systems due to Ehrenfeucht and Rozenberg [21, 22].

Given an initialized transition system A isomorphic to $RG(SN(A))$, the axioms should explain the structure of A in terms of the net system $SN(A)$, since precisely, up to an isomorphism, A is generated by applying inductively the net firing rule from the initial marking of $SN(A)$. In order that the axioms can be independent of the SN operator, they should not refer directly to the places of $SN(A)$, nor to the instanciations of the firing rule to the net $SN(A)$. However, because the places of $SN(A)$ and the regions of A are in bijective correspondence, the regions of A may be used in the axioms as a substitute for the places of $SN(A)$.

Assume $A \cong RG(SN(A))$. In $RG(SN(A))$, two different markings M and M' must differ at some place p. Two different states s and s' of A, thus associated by the isomorphism to different markings M and M' of $SN(A)$, must therefore differ on membership to the region p of A (the places of $SN(A)$ are the regions of A). In $RG(SN(A))$, an event e is disabled at marking M if and only if some place p in the preset of e in $SN(A)$ is not marked in M, or some place p' in the postset of e in $SN(A)$ is marked in M. In A, whenever an event e is disabled at some state s, thus associated by the isomorphism to some marking M of $SN(A)$, it must be the case that either s fails to belong to some region p in the preset of e, or s already belongs to some region p' in the postset of e. A stronger form of the latter axiom, ignoring the second member of this alternative, was used in [28].

As a result, the characterization of quasi-elementary transition systems is comprised of two separation axioms, called State Separation and Event-State Separation. Both axioms are graph-theoretic, but they are not first order since they use regions which are a second order concept (regions are sets). As a matter of fact, a first order graph theoretic characterization of elementary (or quasi-elementary) transition systems is missing.

An important outcome of the axiomatic characterization of elementary (or quasi-elementary) transition systems is to lead to net synthesis algorithm that differ from the brute force algorithm suggested in the end of Sec. 1.3, and that yield nets with size polynomial (instead of exponential) in the size of the transition system taken as input. Given an initialized transition system A, using the terminology of [19, 20], call an *admissible set of regions* any subset of regions $R \subseteq R(A)$ such that $A \cong RG(SN_R(A))$. Then, a set of regions R is admissible if and only if it contains witnesses for the satisfaction of all instances of the separation axioms in A, and the global number of these instances is polynomial in the size of A. Therefore, if an initialized transition system A has some admissible set of regions, then $A \cong RG(SN_R(A))$ for some set of regions R with size polynomial in the size of A. Unfortunately, to check the separation axioms, one cannot dispense with computing the set $R(A)$ of all regions of A, which may have size exponential in the size of A.

Regions and the two separation axioms are the corner stones of the theory and of all algorithms presented in this book.

Let us now turn to statements.

Definition 1.38. *Given an initialized transition system $A = (S, E, \delta, s_0)$, a subset of regions $R \subseteq R(A)$ is* admissible *if $A \cong RG(N)$ for $N = SN_R(A)$, i.e., A is isomorphic to the reachability graph of the net system synthesized from R.*

Example 1.39. The initialized transition system A shown on the left of Fig. 1.17 is isomorphic to the reachability graph of the net system $N = SN_R(A)$ shown on the right of this figure, where $R = \{n_1, h_1, c_1, f, n_2, h_2, c_2\}$ is the admissible set of regions of A defined by the columns of the table displayed immediately after the figure.

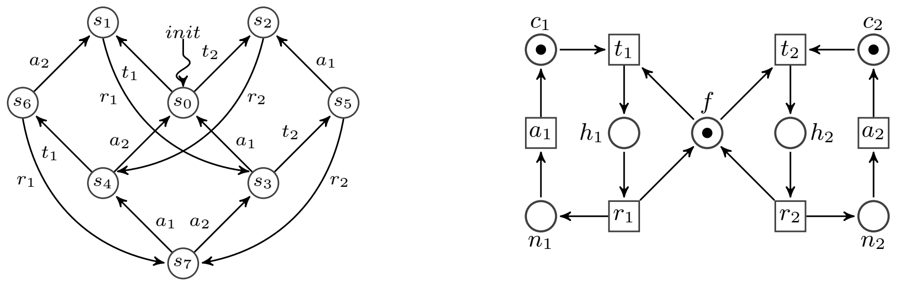


Fig. 1.17. an initialized transition system with an elementary net realization

$\in$	n_1	h_1	c_1	f	c_2	h_2	n_2
s_0	0	0	1	1	1	0	0
s_1	0	1	0	0	1	0	0
s_2	0	0	1	0	0	1	0
s_3	1	0	0	1	1	0	0
s_4	0	0	1	1	0	0	1
s_5	1	0	0	0	0	1	0
s_6	0	1	0	0	0	0	1
s_7	1	0	0	1	0	0	1

The rows of the table give the isomorphism φ between A and $RG(N)$, e.g., $\varphi(s_0) = \{c_1, f, c_2\}$. For every state s and region $r \in R$, $s \in r \Leftrightarrow r \in \varphi(s)$. In other words, $\varphi(s) = \{r \in R \mid s \in r\}$, i.e., φ represents each state by the subset of regions $r \in R$ which it belongs to. In view of Def. 1.20, one can moreover extract from the above table a representation of the events by ordered symmetric differences between representations of states. From $s_4 \xrightarrow{t_1} s_6$ for instance, we get that ${}^\bullet t_1 = \varphi(s_4) \setminus \varphi(s_6) = \{c_1, f\}$ and $t_1{}^\bullet = \varphi(s_6) \setminus \varphi(s_4) = \{h_1\}$. The injectivity of the map φ defined by $\varphi(s) = \{r \in R \mid s \in r\}$ is a necessary condition but not a sufficient condition for a subset $R \subseteq R(A)$ to be an admissible set of regions. The necessary and sufficient condition is stated in the next proposition. □

Proposition 1.40. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system and let $R \subseteq R(A)$ be a subset of regions. Then R is an admissible set of regions if and only if the following two properties are satisfied (where $R_s = \{r \in R \mid s \in r\}$):*

1. *$\forall s, s' \in S \quad R_s = R_{s'} \Rightarrow s = s'$.*
2. *$\forall s \in S\ \forall e \in E \quad ({}^\circ e \subseteq R_s \wedge e^\circ \cap R_s = \emptyset) \Rightarrow s \xrightarrow{e}$*

Proof. By definition of $SN_R(A)$, the initial marking of this net system is $M_0 = R_{s_0}$. As $SN_R(A)$ is a restriction of $SN(A)$, by Lemma 1.33, $s \xrightarrow{e} s' \Rightarrow R_s[e\rangle R_{s'}$ for any transition of A. Let φ be the map defined by $\varphi(s) = R_s$. Since all states of A (respectively all markings of $SN(A)$) may be reached inductively from s_0 (respectively from M_0) and since both transition systems A and $RG(SN_R(A))$ are deterministic, φ is an isomorphism between A and $RG(SN_R(A))$ if and only if it is an injection ($\forall s, s' \in S \quad R_s = R_{s'} \Rightarrow s = s'$) and for any state s, if an event e can fire in R_s, then this event e is enabled in s, i.e. $\forall s \in S\ \forall e \in E \quad ({}^\circ e \subseteq R_s \wedge e^\circ \cap R_s = \emptyset) \Rightarrow s \xrightarrow{e}$. □

Corollary 1.41. *An elementary or quasi-elementary transition system $A = (S, E, \delta, s_0)$ always has a net realization $N = (P, E, F, M_0)$ with number of places $|P|$ less than or equal to $|S| \times ((|S| - 1)/2 + |E|)$.* □

Definition 1.42. *Given an initialized transition system $A = (S, E, \delta, s_0)$, we say that a region $r \in R(A)$* separates *two states s and s' of A if $s \in r \wedge s' \notin r$*

or $s \notin r \wedge s' \in r$. *We say that a region* $r \in R(A)$ separates *an event* e *from a state* s *of* A *if either* $r \in {}^\circ e \wedge s \notin r$ *or* $r \in e^\circ \wedge s \in r$. *We say that a region* $r \in R(A)$ separates strongly *an event* e *from a state* s *of* A *if* $r \in {}^\circ e \wedge s \notin r$.
♢

Consider the following three properties of A and $R \subseteq R(A)$:

- State Separation Property (SSP)

$$\forall s, s' \in S \quad s \neq s' \Rightarrow \exists r \in R \quad r \text{ separates } s \text{ and } s'$$

- Event-State Separation Property (ESSP)

$$\forall e \in E \; \forall s \in S \quad \neg(s \xrightarrow{e}) \Rightarrow \exists r \in R \quad r \text{ separates } e \text{ from } s$$

- Strong Event-State Separation Property (SESSP)

$$\forall e \in E \; \forall s \in S \quad \neg(s \xrightarrow{e}) \Rightarrow \exists r \in R \quad r \text{ separates strongly } e \text{ from } s$$

The main results of the section are the following two theorems.

Theorem 1.43. *Let* $A = (S, E, \delta, s_0)$ *be a loop-free and simple initialized transition system. A is an elementary transition system if and only the properties SSP and ESSP are satisfied for the set* $R = R(A)$ *of all regions of A. For any subset of regions* $R \subseteq R(A)$, *the net system* $N = SN_R(A)$ *is an elementary net realization of A if and only the properties SSP and ESSP are satisfied for R. Further, the net system* $SN_R(A)$ *is a contact-free net realization of A if and only the properties SSP and SESSP are satisfied for R.*

Proof. The first assertion follows from Prop. 1.31, Th. 1.37 and Prop. 1.40. The second assertion follows from Prop. 1.31 and 1.40. The third assertion follows from the fact that the complement $S \setminus r$ of a region $r \in R(A)$ is a region $\overline{r} \in R(A)$ with ${}^\circ\overline{r} = r^\circ$. □

Theorem 1.44. *Let* $A = (S, E, \delta, s_0)$ *be an initialized transition system. A is a quasi-elementary transition system if and only the properties SSP and ESSP are satisfied for the set* $R = R(A)$ *of all regions of A. For any subset of regions* $R \subseteq R(A)$, *the net system* $SN_R(A)$ *is a quasi-elementary net realization of A if and only the properties SSP and ESSP are satisfied for R.*

Proof. Similar to the proof of Th. 1.43. □

Example 1.45. Fig. 1.18 shows two initialized transition systems in which states s_1 and s_2 cannot be separated by any region. Fig. 1.19 shows two initialized transition systems in which event c cannot be separated from state s by any region. The verification is left as an exercice (Exer. 1.4). □

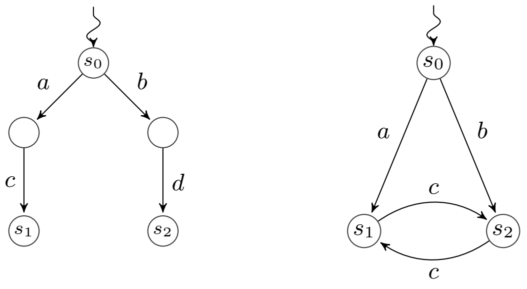


Fig. 1.18. two transition systems where State Separation fails

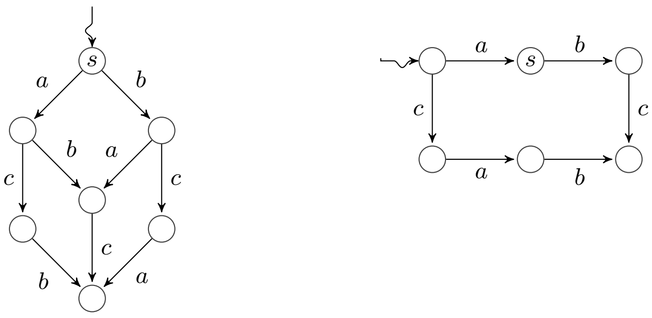


Fig. 1.19. two transition systems where Event-State Separation fails

Definition 1.46. *We say that the properties SSP, ESSP and SESSP are satisfied in A if they are satisfied for $R = R(A)$, and we let $SSP(R)$, $ESSP(R)$ and $SESSP(R)$ mean the properties SSP, ESSP and SESSP, respectively, with a given parameter R. An initialized transition system A is said to be* separated *if the properties SSP and ESSP are satisfied in A. A set of regions $R \subseteq R(A)$ is* strongly admissible *if the properties $SSP(R)$ and $SESSP(R)$ are satisfied.* $\diamondsuit$

Remark 1.47. A is separated if and only if $R(A)$ is admissible (Def. 1.38), and then indeed, $R(A)$ is strongly admissible, because every region has a complementary region. In view of Theorems 1.43 and 1.44, a set of regions $R \subseteq R(A)$ is admissible if and only if the properties $SSP(R)$ and $ESSP(R)$ are satisfied. $\square$

We have obtained a joint characterization of the classes of elementary transition systems or quasi-elementary transition systems in terms of separation axioms. The axiomatization, which relies on regions, does not lead to efficient decision and realization procedures for the basic net synthesis problem. Basically, given a loop-free and simple, resp. arbitrary, initialized transition system A, in order to decide whether A has some elementary, resp. quasi-elementary, net realization, one should decide for every state s whether there exist regions separating s from all other states s' and from all events e disabled at s. The difficulty is to check these properties without constructing all regions in $R(A)$, specially when no net realization exists.

Example 1.48. Consider the four season transition system shown in Fig. 1.20. Each event labels a unique transition, hence every subset of states is a region.

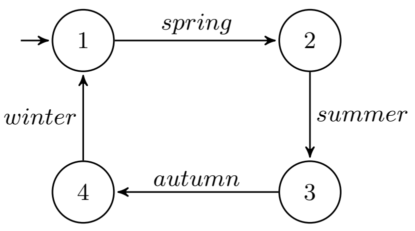


Fig. 1.20. the four seasons transition system

The net system synthesized from all regions except the trivial regions is shown in Fig.1.21. $R_4 = \{\{1\},\{2\},\{3\},\{4\}\}$ is a strongly admissible set of regions.

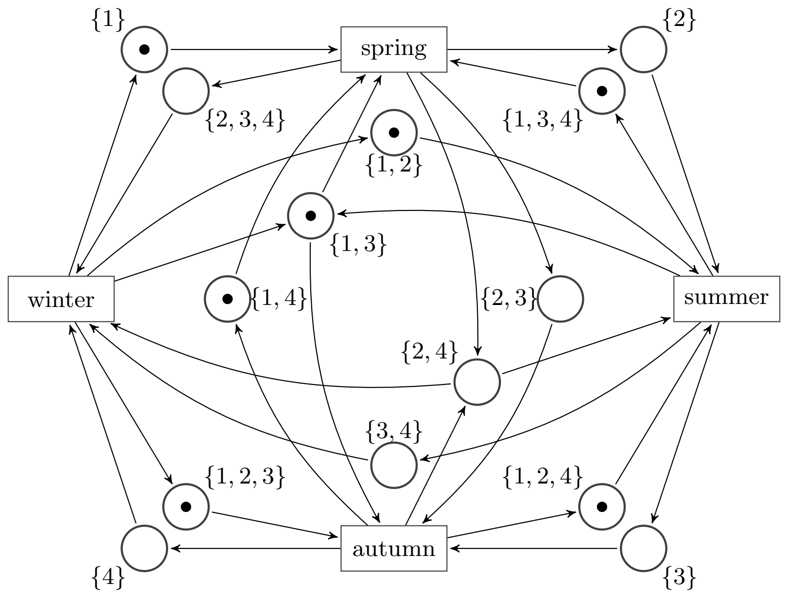
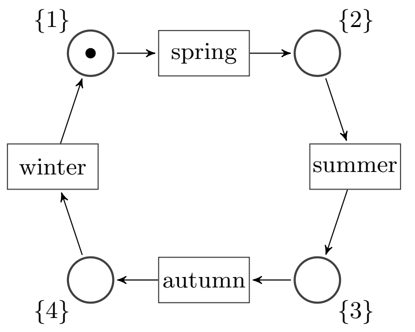


Fig. 1.21. net synthesized from the four seasons transition system

$R_3 = \{\{1,3\},\{2,3\},\{3,4\}\}$ is an admissible set of regions. The net systems N_4 and N_3 synthesized from R_4 and R_3 are shown in Fig. 1.22. N_4 is contact-free, while N_3 has (many) contact situations. R_3 and R_4 are two minimal sets



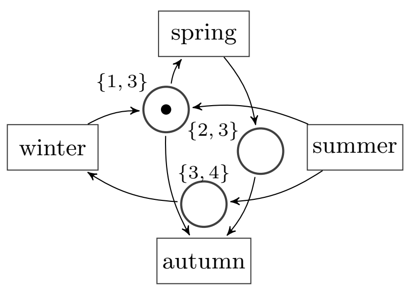


Fig. 1.22. two incomparable minimal sets of admissible regions

of admissible regions, i.e. they have no admissible and strict subset. Adding complementary regions to R_3 leads to a strongly admissible set of regions $R_6 = \{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}\}$. The net system N_6 is shown in Fig. 1.8. R_4 and R_6 are two minimal sets of strongly admissible regions. This example shows that minimal sets of admissible regions may be incomparable (w.r.t. set inclusion) and may even have different cardinalities (R_4 and R_3). The situation is similar for minimal sets of strongly admissible regions (e.g., R_4 and R_6). □

As a concluding observation, we would like to underline that the State Separation axiom is reminiscent of the axiom used in Birkhoff duality to identify finite atomic Boolean algebras that may be represented as powersets of a set. The connection between atomic Boolean algebras and sets is comprised of two operators. One operator $PFilt$ maps each proposition of a Boolean algebra to the set of prime filters that contain this proposition. Recall that a filter is an upwards closed set of propositions which is also closed under binary meets. The other operator $Bool$ maps each subset of a set S identically to itself in 2^B, the Boolean algebra of subsets of S equipped with union, intersection and complementation. A finite atomic Boolean algebra B may be represented as the powerset of a set if and only if $B \cong Bool(PFilt(B))$, if and only if any two different propositions b and b' are separated by some prime filter f such that $b \in f \Leftrightarrow b' \notin f$. The connection between initialized transition systems and net systems constructed in this book presents analogies with this classical connection between Boolean algebras and sets. States of transition systems play the role of boolean propositions and regions of transition systems play the role of prime filters. Now, Boolean algebras and sets are essentially static entities, while transition systems and net systems belong to the realm of dynamic systems. This explains why a second separation axiom is needed in our case, namely the Event-State Separation axiom which has no counterpart in Birkhoff duality.

1.5 Canonical Net Versions Yield Optimal Realizations

The connection between initialized transition systems and net systems built up in sections 1.2 and 1.3 is comprised of two operators, the Reachability Graph and Synthesized Net construction operators (RG and SN). The set of initialized transition systems A such that $A \cong RG(SN(A))$ defines the kernel of this connection on the side of transition systems. In Sec. 1.4, we have characterized this kernel by two separation axioms. Initialized transition systems in which both axioms are satisfied have *exact* net realizations. Initialized transition systems in which one or the other separation axiom fails to be satisfied can only be given *approximate* net realizations. We will show in this section that the approximate net realizations provided by the SN operator are *optimal* in a quite precise sense. For this purpose, we show that the operators RG and SN establish a Galois connection between initialized transition systems and net systems, i.e., that $A \leq RG(N) \Leftrightarrow N \leq SN(A)$ for adequate preorder relations.

Instead of a Galois connection, we could have established a dual adjunction between initialized transition systems and net systems in the spirit of [28] or [3]. However, dealing with the issue of optimality is easier in an order-theoretic setting than in a categorical setting. We have therefore preferred Galois connections, that require light machinery and are moreover well-known from computer scientists since they form the basis of abtract interpretation.

The development given in the rest of the section targets three objectives. First, it provides optimality results needed in subsequent sections of the chapter, where we consider alternative statements of the net synthesis problem. Second, it provides several different but equivalent views of regions, needed in the next chapter on elementary or quasi-elementary net synthesis algorithms. Last but not least, it gives a flavor, on the simple and concrete case of quasi-elementary nets, of the unified theory of regions that will be presented in Part II of this book for arbitrary types of nets.

Let us now enter the technical development. Recall that by *initialized transition system*, we mean *deterministic, reachable and reduced* initialized transition systems (Def. 1.30), while by *net system*, we mean *quasi-elementary net system* (Def. 1.23), thus allowing isolated or equivalent transitions, but banning out equivalent places and dead transitions. The first step towards proving that the operators RG and SN establish a Galois connection between initialized transition systems and net systems is to equip them with preorder relations.

Definition 1.49. *Given two initialized transition systems* $A = (S, E, \delta, s_0)$ *and* $A' = (S', E, \delta', s'_0)$ with the same set of events*, let* $A \leq A'$ *if there exists a map* $\varphi : S \to S'$*, called a* simulation*, such that* $\varphi(s_0) = s'_0$ *and* $s \xrightarrow{e} s' \Rightarrow \varphi(s) \xrightarrow{e} \varphi(s')$ *for all states* $s, s' \in S$ *and events* $e \in E$. ◇

When a simulation map φ exists, it is necessarily unique, and we say that it *justifies* the simulation relation $A \leq A'$. This relation is a preorder, i.e. it is reflexive and transitive. If $A \leq A'$ and $A' \leq A$, then the two simulation maps $\varphi : S \to S'$ and $\varphi' : S' \to S$ are inverse to each other, hence in this case A and A' are isomorphic ($A \cong A'$).

Definition 1.50. *Given two net systems $N = (P, T, F, M_0)$ and $N' = (P', T, F', M_0')$ with the same set of transitions, let $N \leq N'$ if there exists a map $\iota : P \to P'$, called an* embedding *of N into N', such that for all $p \in P$, $M_0(p) = M_0'(\iota(p))$, ${}^\bullet p = {}^\bullet\iota(p)$ and $p^\bullet = \iota(p)^\bullet$.* $\diamondsuit$

When an embedding map ι exists, it is necessarily injective (because N has no equivalent places) and unique (because N' has no equivalent places), and we say that it *justifies* the relation $N \leq N'$. If $N \leq N'$ and $N' \leq N$, then the embedding maps $\iota : P \to P'$ and $\iota' : P' \to P$ are inverse to each other, hence in this case N and N' are isomorphic.

Our goal is to establish the following statement and to examine its implications.

Theorem 1.51. *For any initialized transition system A with set of events E and for any net system with the set of transitions $T = E$,*

$$A \leq RG(N) \Leftrightarrow N \leq SN(A)$$

Th. 1.51 states the existence of a Galois connection between initialized transition systems and net systems. In order to establish this theorem, we need to come back to the notion of regions and observe that they may be presented in many different disguises.

Remark 1.52. A region r of an initialized transition system $A = (S, E, \delta, s_0)$ may be specified equivalently in the following ways:

1. explicitly, as a set of states $r \subseteq S$;
2. by its characteristic function $r : S \to \{0, 1\}$, i.e., $r(s) = 1$ if $s \in r$, $r(s) = 0$ otherwise;
3. by a function $r : \{init\} \cup E \to \{-1, 0, 1\}$, such that $r(init) = r(s_0) \in \{0, 1\}$ and $r(e) = r(s') - r(s)$ for any transition $s \xrightarrow{e} s'$ in A. This function is called the *signature* of the region r. The preset and postset of the region r may then be retrieved as ${}^\circ r = \{e \in E \mid r(e) = 1\}$ and $r^\circ = \{e \in E \mid r(e) = -1\}$;
4. by a function $r : S \cup E \to \{-1, 0, 1\}$, such that $r(s) \in \{0, 1\}$ for all $s \in S$ and $r(e) = r(s') - r(s)$ for any transition $s \xrightarrow{e} s'$ in A. $\square$

Remark 1.53. Given $A = (S, E, \delta, s_0)$, a map $f : S \to \{0, 1\}$ is a region of A if and only it has a *companion map* $g : E \to \{-1, 0, 1\}$ such that

$$s \xrightarrow{e} s' \quad \Rightarrow \quad f(s') = f(s) + g(e)$$

Such a map g, when it exists, is unique. Conversely, a map $g : E \to \{-1, 0, 1\}$ which is not the constant map $g = 0$ has at most one companion map f

(because every event is enabled in some state of A and every state of A can be reached from s_0). The constant map $g = 0$ has two companion maps, namely the characteristic functions of the two trivial regions. □

Example 1.54. Consider again the initialized transition system in Fig. 1.23. Five forms defining the same region of this initialized transition system are:

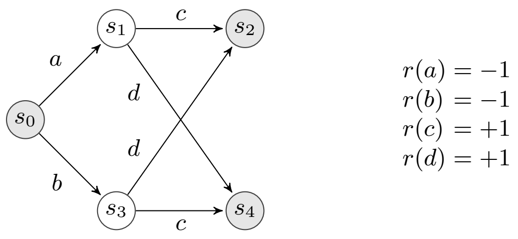


Fig. 1.23. various representations of a region in a transition system

1. $r = \{s_0, s_2, s_4\}$;
2. $r(s_0) = r(s_2) = r(s_4) = 1$, $r(s_1) = r(s_3) = 0$;
3. $r(init) = 1$, $r(a) = r(b) = -1$, $r(c) = r(d) = +1$;
4. $r(s_0) = r(s_2) = r(s_4) = 1$, $r(s_1) = r(s_3) = 0$, $r(a) = r(b) = -1$, $r(c) = r(d) = +1$;
5. $r = f$ where $f(s_0) = f(s_2) = f(s_4) = 1$, $f(s_1) = f(s_3) = 0$ and the companion map of f is $g(a) = g(b) = -1$, $g(c) = g(d) = +1$.

□

Definition 1.55. *The* signature *of a place p of a net system N is the signature of the region $[\![p]\!]$ of $RG(N)$ defined by the extension of this place, i.e., $[\![p]\!](init) = M_0(p)$, $[\![p]\!](e) = -1$ if $e \in p^\bullet$, $[\![p]\!](e) = 1$ if $e \in {}^\bullet p$, and $[\![p]\!](e) = 0$ otherwise.* ◇

The following lemma is crucial to the proof of Th. 1.51.

Lemma 1.56. *Let $A = (S, E, \delta, s_0)$ and $A' = (S', E, \delta', s'_0)$ be initialized transition systems with the same set of events E. Let $A \leq A'$ and let $\varphi : S \to S'$ be the simulation map justifying this relation. Then, for every region r' of A', $r = \varphi^{-1}(r')$ is a region of A. Moreover, r and r' have the same signature (defined in Remark 1.52).*

Proof. If r' is a trivial region of A', then $\varphi^{-1}(r')$ is a trivial region of A. In the converse case, let $r' : \{init\} \cup E \to \{-1, 0, 1\}$ be the signature of the region r', and let $g' : E \to \{-1, 0, 1\}$ be the restriction of this signature on the set of events E. As r' is a region, g' has a companion map $f' : S' \to \{0, 1\}$. Define $f : S \to \{0, 1\}$ and $g : E \to \{-1, 0, 1\}$ by setting $f(s) = f'(\varphi(s))$ and

$g(e) = g'(e)$ for all $e \in E$. As f' and g' are companion maps, f and g are also companion maps, because $s_1 \xrightarrow{e} s_2 \Rightarrow \varphi(s_1) \xrightarrow{e} \varphi(s_2)$ for all states $s_1, s_2 \in S$ and events $e \in E$. Therefore, $r = f^{-1}\{1\} = \varphi^{-1}(f'^{-1}(1)) = \varphi^{-1}(r')$ is a region of A. Finally, $r(init) = r(s_0) = f(s_0) = f'(\varphi(s_0)) = f'(s'_0) = r'(s'_0) = r'(init)$, and $r(e) = g(e) = g'(e) = r'(e)$ for all $e \in E$, hence r and r' have the same signature. □

Th. 1.51 is established by the following two propositions.

Proposition 1.57. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system and let $N = (P, E, F, M_0)$ be a net system. If $A \leq RG(N)$ then $N \leq SN(A)$.*

Proof. Let $\varphi : S \to RS(N)$ be the simulation map justifying the relation $A \leq RG(N)$. By Prop. 1.19, for every place p of N, the extension $[\![p]\!]$ of this place in $RG(N)$ is a region with the same signature as p. By Lemma 1.56, $\varphi^{-1}[\![p]\!]$ is a region of A with the same signature as $[\![p]\!]$. Therefore, the map $\iota : P \to R(A)$ defined by $\iota(p) = \varphi^{-1}[\![p]\!]$ for all $p \in P$ is an embedding map that justifies the relation $N \leq SN(A)$. □

Proposition 1.58. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system and let $N = (P, E, F, M_0)$ be a net system. If $N \leq SN(A)$ then $A \leq RG(N)$.*

Proof. Let $\iota : P \to R(A)$ be the embedding map that justifies the relation $N \leq SN(A)$ (recall that the places of $SN(A)$ are the regions of A). For any state $s \in S$, let $\varphi(s)$ be the marking of N defined by $p \in \varphi(s) \Leftrightarrow s \in \iota(p)$. By Def. 1.25 and 1.50, $p \in M_0$ if and only if $s_0 \in \iota(p)$, hence $p \in M_0$ if and only if $p \in \varphi(s_0)$, i.e., $\varphi(s_0) = M_0$. In order to show that φ is a simulation map justifying the relation $A \leq RG(N)$, it remains to show that $\varphi(s) \in RS(N)$ for all $s \in S$, and that $s \xrightarrow{e} s'$ in A entails $\varphi(s)[e\rangle\varphi(s')$ in N for all $e \in E$. As $\varphi(s_0) = M_0$ and every state of A can be reached inductively from s_0, it is sufficient to prove that $s \xrightarrow{e} s'$ entails $\varphi(s)[e\rangle\varphi(s')$. By Lemma 1.33, $s \xrightarrow{e} s'$ entails $R_s[e\rangle R_{s'}$ in $SN(A)$. Consider an arbitrary place p of N, then $r = \iota(p)$ is a region of A, and by Remark 1.53, $r(s') = r(s) + r(e)$. In view of Def. 1.50, p and $\iota(p)$ have the same signature, hence $r(s') = r(s) + F(e,p) - F(p,e)$ where F is the flow relation of N. Now $r(s) = 1$ iff $s \in \iota(p)$ iff $p \in \varphi(s)$, and similarly, $r(s') = 1$ iff $p \in \varphi(s')$. Therefore, $\varphi(s')(p) = \varphi(s)(p) + F(e,p) - F(p,e)$. As p was chosen arbitrarily, it follows that $\varphi(s)[e\rangle\varphi(s')$ in N. □

Notation 1.59 *For convenience, we let A^{SN} and N^{RG} be equivalent notations for $SN(A)$ and $RG(N)$, respectively. We let operators SN and RG be composed from left to right in superscripts, e.g., $A^{SN \cdot RG}$ means $RG(SN(A))$.*

We enumerate below some direct but important consequences of the Galois connection between initialized transition systems and net systems stated in Th.1.51 ($A \leq N^{RG} \Leftrightarrow N \leq A^{SN}$ in the above notation). All properties listed are classical properties of Galois connections, reinterpreted in the specific setting of transition systems and net systems. When symmetric pairs of properties of transition systems and net systems are stated simultaneously, we sketch the proof of the properties for net systems (the proofs of the similar properties for transition systems follow by exchanging A and N, and RG and SN).

Property 1.60. $A \leq A^{SN \cdot RG}$ and $N \leq N^{RG \cdot SN}$.

Proof. $N \leq N^{RG \cdot SN} \Leftrightarrow N^{RG} \leq N^{RG}$. □

Interpretation: every initialized transition system A is simulated by the reachability graph of its canonical net version $SN(A)$, and every net system N is isomorphic to a restriction of the net system synthesized from its reachability graph $RG(N)$.

Property 1.61. $A_1 \leq A_2 \Rightarrow A_2^{SN} \leq A_1^{SN}$ and $N_1 \leq N_2 \Rightarrow N_2^{RG} \leq N_1^{RG}$.

Proof. $N_1 \leq N_2 \leq N_2^{RG \cdot SN} \Rightarrow N_2^{RG} \leq N_1^{RG}$. □

Interpretation: Both operators RG and SN are decreasing: increasing the set of places of a net decreases its behaviour; increasing a transition system decreases the set of possible signatures of net places compatible with this behaviour.

Property 1.62. $A^{SN \cdot RG}$ is the best upper approximation of A (up to an isomorphism) by the reachability graph of a net system.

Proof. $A \leq A^{SN \cdot RG}$, and $A \leq N^{RG} \Rightarrow N \leq A^{SN} \Rightarrow A^{SN \cdot RG} \leq N^{RG}$. □

Interpretation: The canonical net version $SN(A)$ of A is *optimal* amongst the approximate net realization of A, even if A is not a quasi-elementary transition system.

Property 1.63. A^{SN} and $A^{SN \cdot RG \cdot SN}$, respectively N^{RG} and $N^{RG \cdot SN \cdot RG}$, are isomorphic.

Proof. $N^{RG} \leq N^{RG \cdot SN \cdot RG}$, and $N \leq N^{RG \cdot SN} \Rightarrow N^{RG \cdot SN \cdot RG} \leq N^{RG}$.
□

Interpretation: The best upper approximation of A by the reachability graph of a net system is the reachability graph $A^{SN \cdot RG}$ of A^{SN}, and iterating net synthesis from $A^{SN \cdot RG}$ is useless since it yields again A^{SN} up to an isomorphism.

Definition 1.64. *A net system N is* saturated *if no place can be added to this net system without modifying its behaviour $RG(N)$ (considered up to isomorphisms of graphs), i.e., if $(N \leq N' \ \wedge \ N^{RG} \cong N'^{RG}) \Rightarrow N \cong N'$.* ◇

Property 1.65. If N is saturated, then $N \cong N^{RG \cdot SN}$.

Proof. If one lets $N' = N^{RG \cdot SN}$ in Def. 1.64, then from $N \leq N^{RG \cdot SN}$ (Prop. 1.60) and $N^{RG} \cong N^{RG \cdot SN \cdot RG}$ (Prop. 1.63) it follows that N and $N^{RG \cdot SN}$ are isomorphic. □

Property 1.66. $N^{RG \cdot SN}$ is a saturated net system.

Proof. Assume that $N^{RG \cdot SN} \leq N'$ and $N^{RG \cdot SN \cdot RG} \cong N'^{RG}$, then $N^{RG} \cong N^{RG \cdot SN \cdot RG} \cong N'^{RG} \Rightarrow N^{RG} \leq N'^{RG} \Rightarrow N' \leq N^{RG \cdot SN}$, hence N' and $N^{RG \cdot SN}$ are isomorphic. □

Interpretation: N is saturated if and only if N and $N^{RG \cdot SN}$ are isomorphic. Moreover, since A^{SN} and $A^{SN \cdot RG \cdot SN}$ are isomorphic, the canonical net version A^{SN} of an initialized transition system is saturated, even if A is not a quasi-elementary transition system.

Property 1.67. $A \cong N^{RG}$ for some N if and only if $A \cong A^{SN \cdot RG}$.

Proof. $A \leq A^{SN \cdot RG}$; $A \leq N^{RG} \Rightarrow N \leq A^{SN} \Rightarrow A^{SN \cdot RG} \leq N^{RG} \cong A$. □

Interpretation: This is just a different proof of Th. 1.37.
By Th. 1.37 and Prop. 1.40, $A \cong A^{SN \cdot RG}$ if and only if A is separated (Def. 1.46). As a result, there exists a strong relationship between *separated* initialized transition systems (with admissible sets of regions) and *saturated* net systems (with maximal sets of places). If A is separated, then $A \cong A^{SN \cdot RG}$, hence A may be reconstructed up to an isomorphism from its canonical net version A^{SN}, and A^{SN} is saturated since it is isomorphic to $A^{SN \cdot RG \cdot SN}$. If N is saturated, then $N \cong N^{RG \cdot SN}$, hence N may be reconstructed up to an isomorphism from its reachability graph N^{RG}, and N^{RG} is separated since it is isomorphic to $N^{RG \cdot SN \cdot RG}$. Whenever A is loop-free, simple and separated, separatedness entails that $A \cong A^{SN \cdot RG}$, hence A^{SN} is an elementary net system (by Prop. 1.31). Whenever N is an elementary net system, N^{RG} is loop-free and simple, hence N^{RG} is an elementary transition system (by Prop. 1.31). Therefore, one can state the following result.

Theorem 1.68. *The operators RG and SN restrict to reciprocal bijections between isomorphism classes of separated initialized transition systems and isomorphism classes of saturated net systems. They further restrict to reciprocal bijections between isomorphism classes of elementary transition systems and isomorphism classes of elementary net systems.* □

However, the Galois connection which we have built up in this section cannot be specialized into a Galois connection between loop-free and simple initialized transition systems and elementary net systems.

For instance, as we have already discussed, the initialized transition system A shown in Fig. 1.15 (on page 22) is loop-free and simple, but the transition c is isolated in $SN(A)$ while the transitions a and b are equivalent in $SN(A)$. Searching for an approximate but optimal realization of the considered transition system by an elementary net system would simply make no sense. So, the constraints of simpleness and loop-freeness set on elementary transition systems make no difficulties when the problem dealt with is the *exact* realization of a transition system by a net system, but they become a real obstacle when the problem dealt with is the *approximate* realization of a transition system by a net system, which has at least equal importance in practice.

Remark 1.69. The isomorphism $A^{SN} \cong A^{SN \cdot RG \cdot SN}$ shows that the regions of A and $A^{SN \cdot RG}$ have the same set of signatures. By Lemma 1.56, if φ is the simulation map justifying the relation $A \leq A^{SN \cdot RG}$, then for every region r' of $A^{SN \cdot RG}$, $r = \varphi^{-1}(r')$ is a region of A with the same signature. Therefore, by Prop. 1.19, the set of regions of A is exactly the set of inverse images $\varphi^{-1}(r')$ of the regions r' of $A^{SN \cdot RG}$. □

1.6 Relaxing the State Separation Property

As we saw in Section 1.4, solving the basic net synthesis problem requires that one checks A (the initialized transition system taken as input) for two separation properties (SSP and ESSP) whose conjunction entails $A \cong RG(N)$ for $N = SN(A)$. Sometimes, the primary objective of the synthesis is to construct an elementary net system N with the same language as A but without requiring that A and $RG(N)$ should be isomorphic. This problem will be considered in full generality in Section 1.7. In this section, we want to examine what happens when just dismissing the state separation property (SSP) and building up a net system $N = SN_R(A)$ from any set of regions R of the given transition system A witnessing for the satisfaction of the event-state separation property (ESSP). Dismissing SSP while maintaining ESSP reflects the desire to save on places of the synthesized nets while still conforming to the behaviour of A. Indeed, as we shall see, the axiom ESSP is satisfied in A if and only if $L(A) = L(N)$ and $A \leq RG(N)$ for some net system N, if and only if $L(A) = L(SN_R(A))$ for some set of regions R of A (where $L(A)$ and $L(N)$ are the languages of A and N). Technically speaking, the problem is to synthesize a net system realizing the *quotient* of the transition system A by some equivalence on states compatible with labelled transitions. We examine this problem first for quasi-elementary net systems and next for elementary net sytems. Let us recall some classical definitions and results about quotients.

Definition 1.70. *Given an initialized transition system $A = (S, E, \delta, s_0)$, an equivalence relation $\equiv$ on the set of states S is said to be* compatible with labelled transitions *if $s_1 \xrightarrow{e} s_2$ and $s_1 \equiv s_1'$ entail $s_1' \xrightarrow{e} s_2'$ and $s_2 \equiv s_2'$ for* *some state s'_2. Given an equivalence relation $\equiv$ on S compatible with labelled transitions, the* quotient *of A by $\equiv$ (notation: $(A/\equiv)$) is the initialized transition system $((S/\equiv), E, (\delta/\equiv), [s_0])$ where $S/\equiv$ is the set of equivalence classes $[s]$ of states $s \in S$, $[s_0]$ is the equivalence class of the initial state and $(\delta/\equiv)([s], e) = [\delta(s, e)]$ for every state $s \in S$.* ♢

If $\equiv$ is an equivalence relation on S compatible with labelled transitions, then $s \equiv s'$ entails $L(s) = L(s')$, where $L(s)$ is the language generated by A from state s.

Definition 1.71. *Given $A = (S, E, \delta, s_0)$ and $A' = (S', E, \delta', s'_0)$, a label preserving* morphism of initialized transition systems *$\varphi : A \to A'$ is a map $\varphi : S \to S'$ such that $\varphi(s_0) = s'_0$ and $s \xrightarrow{e} s' \Rightarrow \varphi(s) \xrightarrow{e} \varphi(s')$ for every transition $s \xrightarrow{e} s'$ of A.* ♢

If φ is a label preserving morphism, then for any state $s \in S$, $L(s) \subseteq L(\varphi(s))$. As initialized transition systems are always (assumed to be) deterministic, reachable and reduced, there can exist *at most one* label preserving morphism from A to A'. In the terminology of Section 1.5, the map φ of Def. 1.71 is a simulation map and it justifies the relation $A \leq A'$, which we strengthen to $A \hookrightarrow A'$ in case when the map φ is moreover injective.

Definition 1.72. *A label preserving morphism φ from A to A' is a* saturating morphism *if $\varphi(s) \xrightarrow{e} \; \Rightarrow \; s \xrightarrow{e}$.*

If φ is a saturating morphism from A to A', then for any state $s \in S$, $L(s) \supseteq L(\varphi(s))$, hence $L(s) = L(\varphi(s))$. Therefore, if φ is a saturating morphism, then $L(A) = L(A')$ and whenever $\varphi(s) = \varphi(s')$, states s and s' are language equivalent $(L(s) = L(s'))$.

If $L(A) = L(A')$, then any label preserving morphism from A to A' is a saturating morphism.

As initialized transition systems are always reachable and reduced, saturating morphisms are surjective. Therefore, if a saturating morphism is injective, then it is an isomorphism.

Given $A = (S, E, \delta, s_0)$, for any equivalence relation $\equiv$ on S compatible with labelled transitions, the map $(\cdot/\equiv)$ that sends each state $s \in S$ to its equivalence class $[s]$ is a surjective morphism. Conversely, if φ is a saturating morphism from $A = (S, E, \delta, s_0)$ to $A' = (S', E, \delta', s'_0)$, then the equivalence relation $\equiv$ on S defined as $s \equiv s'$ iff $\varphi(s) = \varphi(s')$ is compatible with labelled transitions and the associated quotient $A/\equiv$ is isomorphic to A'. In view of this, by extension, we say that A' is a *quotient* of A (notation $A \rhd A'$) if there exists a saturating morphism $\varphi : A \to A'$. Equivalently, we say that A may be *folded* to A' and that φ is a *folding* morphism.

Example 1.73. Fig. 1.24 shows two initialized transition systems A (on the left) and A' (on the right) such that A may be folded to A'. The map $\varphi(s_0) = s'_0$ and $\varphi(s_1) = \varphi(s_2) = s'_{1,2}$ is indeed a saturating morphism from A to A'. The equivalence $\equiv$ induced by φ identifies s_1 and s_2 and discriminates all other pairs of states. This equivalence is compatible with labelled transitions, and A' is isomorphic to the quotient $A/\equiv$. $\square$

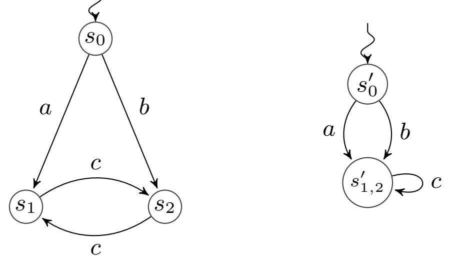


Fig. 1.24. $A \rhd A'$

Example 1.74. Fig. 1.25 shows two initialized transition systems A (on the left) and A' (on the right) such that A cannot be folded to A'. The map $\varphi(s_0) = s'_0$, $\varphi(s_1) = \varphi(s_2) = s'_{1,2}$, $\varphi(s_3) = s'_3$, $\varphi(s_4) = s'_4$ and $\varphi(s_5) = \varphi(s_6) = s'_{5,6}$ is a label preserving morphism, but it is not a saturating morphism since, e.g., $s'_{1,2} \xrightarrow{d}$ but not $s_1 \xrightarrow{d}$. Indeed, φ identifies states s_1 and s_2 which are not language equivalent. $\square$

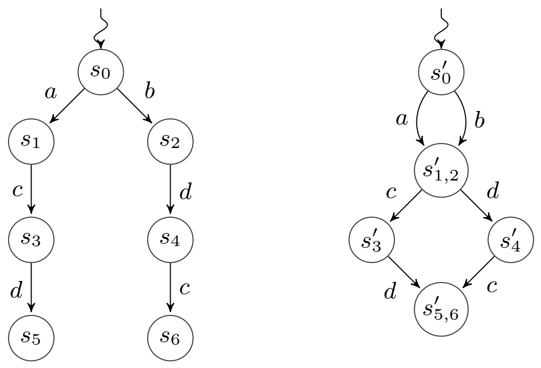


Fig. 1.25. $A \not\rhd A'$

By dismissing the states separation property SSP, the theory of regions may be adapted to the realization of initialized transition systems by net systems up to an equivalence on states compatible with labelled transitions.

Proposition 1.75. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system and let N be a net system with set of transitions $T = E$.*

1. *The following three conditions are equivalent*
   a) $A \rhd RG(N)$,
   b) $A \leq RG(N)$ *and* $L(A) = L(N)$,
   c) $N \cong SN_R(A)$ *for some set of regions* $R \subseteq R(A)$ *such that* $ESSP(R)$ *holds in* A.
2. *The following conditions are equivalent*
   a) $A \hookrightarrow RG(N)$,
   b) $N \cong SN_R(A)$ *for some set of regions* $R \subseteq R(A)$ *such that* $SSP(R)$ *holds in* A.
3. *The following conditions are equivalent*
   a) $A \cong RG(N)$, *i.e.,* N *is a net system realization of* A,
   b) $N \cong SN_R(A)$ *for some set of regions* $R \subseteq R(A)$ *such that* $SSP(R)$ *and* $ESSP(R)$ *hold in* A, *i.e.,* R *is a set of admissible regions of* A.

In order to establish Prop. 1.75, we need introducing some notation, which is done in the next definition.

Definition 1.76. *Let* $\eta_A : A \to RG(SN(A))$ *be the map defined by* $\eta_A(s) = \{r \in R(A) \mid s \in r\}$. *For any set of regions* $R \subseteq R(A)$, *let* $\eta_{A,R} : A \to RG(SN_R(A))$ *be the map defined by* $\eta_{A,R}(s) = R_s = \{r \in R \mid s \in R\}$. $\diamondsuit$

The map η_A represents each state s of A by the marking of the synthesized net $SN(A)$ comprised of the regions which s belongs to. The map η_A is a simulation map and it justifies the relation $A \leq RG(SN(A))$ (see Def. 1.49). The map $\eta_{A,R}$ is the composition of the map $\eta_A : A \to RG(SN(A))$ with the map $\pi_R : RG(SN(A)) \to RG(SN_R(A))$ that projects each marking of $SN(A)$ (the synthesized net) on the subset of places of $SN(A)$ which are defined by regions in R. The map $\eta_{A,R}$ is a simulation map and it justifies the relation $A \leq RG(SN_R(A))$. The three statements in the proposition may now be proven as follows.

Proof. of Prop. 1.75

Ad 1) 1.a) and 1.b) are equivalent by the remarks made after Def. 1.72. 1.b) and 1.c) are equivalent because $A \leq RG(N)$ *iff* $N \leq SN(A)$ (by Th. 1.51) *iff* $N \cong SN_R(A)$ for some set of regions $R \subseteq R(A)$, and $L(A) = L(SN_R(A))$ *iff* $\eta_{A,R}$ is a saturating morphism *iff* ESSP(R) holds in A (by definition of saturating morphisms and in view of the net firing rule applied to the net $SN_R(A)$).

Ad 2) $A \hookrightarrow RG(N)$ *iff* $A \leq RG(N)$ and the label preserving morphism justifying this relation is injective. Now $A \leq RG(N)$ *iff* $N \leq SN(A)$ (by Th. 1.51) *iff* $N \cong SN_R(A)$ for some set of regions $R \subseteq R(A)$, and the label preserving morphism justifying the relation $A \leq RG(SN_R(A))$ is $\eta_{A,R}$. Finally $\eta_{A,R}$ is injective *iff* SSP(R) holds in A.

Ad 3) This restatement of Th. 1.43 follows from the equivalence
$A \cong RG(N) \;\Leftrightarrow\; (A \rhd RG(N) \;\wedge\; A \hookrightarrow RG(N))$. □

The next theorem is the major result of the section. This theorem states that $A^{SN \cdot RG}$ is *optimal* among all quotients $A/\equiv$ of A which may be realized *exactly* by net systems (at least, if event-state separation is satisfied in A, since otherwise such quotients do not exist).

Theorem 1.77. *Let A be an initialized transition system satisfying the event-state separation property. Then $A^{SN \cdot RG}$ is the least* separated *initialized transition system A' such that $A \rhd A'$.*

Proof. By Prop. 1.40 and Th. 1.44, an initialized transition system A' is separated if and only if, for some set of regions $R' \subseteq R(A')$, $A' \cong RG(N')$ for $N' = SN_{R'}(A')$. If $A \rhd A'$, then by Lemma 1.56, for every region r' of A', there exists a region r of A with the same signature. Assume that $A \rhd A'$ and A' is separated. Let $R \subseteq R(A)$ be the set of regions of A with the same signatures as the regions in R'. Then necessarily, $A' \cong RG(N)$ for $N = SN_R(A)$. By assumption, A satisfies the event-state separation property, hence $ESSP(R(A))$ holds in A. By Prop. 1.75, $A \rhd RG(N)$ for $N = SN(A)$. Now $R \subseteq R(A)$ entails $SN_R(A) \leq SN(A)$, and by Property 1.61 following from Th. 1.51, the latter entails $RG(SN(A)) \leq RG(SN_R(A))$. As $RG(SN(A))$ is separated, $RG(SN(A))$ is the least separated initialized transition system A' such that $A \rhd A'$. □

The following example shows that, even though A is a separated initialized transition system, the minimal automaton $Min(A)$ recognizing the language $L(A)$ (see, e.g., [30]) may be non-separated. In particular, $Min(A)$ is not always isomorphic to the least separated folding of A, namely $A^{SN \cdot RG}$. As minimization is a form of folding ($A \rhd Min(A)$), it follows that the class of separated initialized transition systems is *not* closed under folding.

Example 1.78. Fig. 1.26 displays, from left to right, an elementary net system N, its reachability graph $RG(N)$, and the minimal automaton $Min(RG(N))$ which is obtained by identifying the two language equivalent markings $\{p_3\}$ and $\{p_4\}$. On the one hand, $RG(N)$ is separated (by construction). On the other hand, $Min(RG(N))$ is not separated, and since it is not isomorphic to $RG(N)$, it is not isomorphic either to $(RG(N))^{SN \cdot RG}$ (because $RG(N) \cong (RG(N))^{SN \cdot RG}$ by Property 1.63 following from Th. 1.51). In fact, a region of $Min(RG(N))$ contains s_2 if and only if it contains s_3, because $s_2 \overset{cd}{\to} s_6$ and $s_3 \overset{dc}{\to} s_6$ converge (in s_6) and the sequences cd and dc are permutations of each other. Therefore, the states s_2 and s_3 cannot be separated. For the same reason, no region can separate the event d from the state s_2. The signature of the region $[\![p_3]\!]$, that separates d and m_3 from m_2 in $RG(N)$, is lost through the minimization $\varphi : RG(N) \to Min(RG(N))$ (this region is not the inverse image $\varphi^{-1}(r)$ of any region of $Min(RG(N))$). □

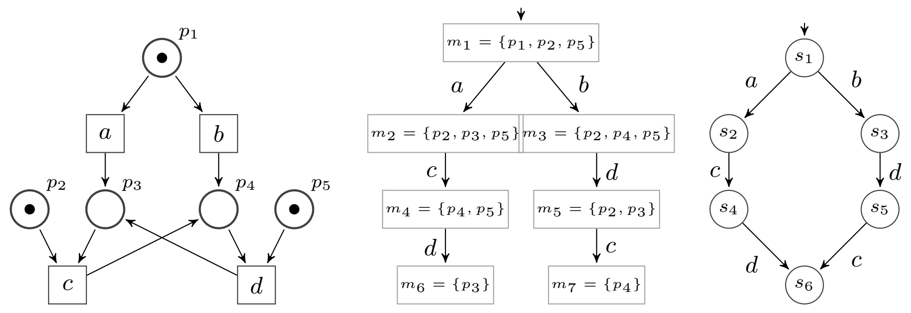


Fig. 1.26. separated transition systems are not closed under folding

Folding is a well-behaved operation that preserves and reflects languages of transition systems, but it may dramatically interfere with net synthesis, as Example 1.78 has shown. In the special case where A is a minimal automaton, $A \rhd A'$ entails $A \cong A'$, and in particular, $A \rhd RG(N)$ entails $A \cong RG(N)$ for any elementary net system N. Therefore, one can state the following corollary of Proposition 1.75.

Corollary 1.79. *A minimal initialized transition system $A \cong Min(A)$ is separated if and only if it satisfies the event state separation property. More precisely, $A \cong RG(N)$ for some net system N if and only if $N \cong SN_R(A)$ and A satisfies $ESSP(R)$ for some set of regions $R \subseteq R(A)$.* □

We want now to address the case where one adds as a specific requirement that N should be an *elementary net system*.

Example 1.80 (Exple. 1.28 continued). To start with an example, consider the initialized transition system A depicted in Fig. 1.13, on page 19. It may be checked that the event-state separation property holds in A (see Examples 1.21 on p. 19, and 1.28 on p. 22). However, for any set of regions R such that $ESSP(R)$ holds in A, the reachability graph $RG(SN_R(A))$ is isomorphic to the initialized transition system depicted on the right of Fig. 1.16, on p. 23 (note that $SN_R(A)$ is necessarily a subnet of the net system depicted on the left of Fig. 1.16). Therefore, the transitions a and b (or c and d) must be equivalent in $SN_R(A)$. □

As the above example shows, it may happen, for an initialized transition system A which is *loop-free and simple*, and where event-state separation holds, that $A \rhd RG(N)$ for no elementary net system N. For any set of regions R enforcing event-state separation, $RG(SN_R(A))$ is indeed a quotient of A, but loops or multiple transitions between states may be produced by folding A to $RG(SN_R(A))$ (using the folding morphisn $\eta_{A,R}$ of Def. 1.76). This unpleasant situation cannot occur if one requires from R to enforce in A both state separation and event-state separation, since in this case $A \cong RG(SN_R(A))$. If state separation is dismissed, then one must compensate for this weakening by adding two new regional axioms, which express the absence of isolated or equivalent transitions in the synthesized net system $SN_R(A)$.

Proposition 1.81. *An initialized transition system A may be folded to some elementary transition system A' if and only if event-state separation is satisfied in A together with the following two properties:*

event-effectiveness (EE): $(\forall e \in E) \quad {}^\circ e \neq \emptyset$
i.e., all events have non empty presets,
event-simpleness (ES): $(\forall e, e' \in E) \quad ({}^\circ e = {}^\circ e' \wedge e^\circ = e'^\circ) \Rightarrow e = e'$.

Proof. Let $A \rhd RG(N)$ for some elementary net system N. By Prop. 1.75 $N \cong SN_R(A)$ for some set of regions $R \subseteq R(A)$ such that A satisfies $ESSP(R)$. Without loss of generality we may assume that N is saturated, i.e., that $R = R(A)$ and $N = SN(A)$. Indeed, seeing that $N^{RG\cdot SN}$ is elementary if N is elementary, one can always replace N with $N^{RG\cdot SN}$ which is saturated and satisfies $A \rhd RG(N^{RG\cdot SN})$, since $RG(N) = N^{RG} \cong N^{RG\cdot SN\cdot RG} = RG(N^{RG\cdot SN})$ (by Property 1.63 following from Th. 1.51). As N is an elementary net system, it has neither isolated transitions $((\forall e \in E) \quad {}^\bullet e \neq \emptyset)$ nor equivalent transitions $((\forall e, e' \in E) \quad ({}^\bullet e = {}^\bullet e' \wedge e^\bullet = e'^\bullet) \Rightarrow e = e')$. For any place $p \in {}^\bullet e$, the extension $r = [\![p]\!]$ of p is a region of $RG(N)$ and $r \in {}^\circ e$ (Prop. 1.19); by Lemma 1.56, $r' = \eta_A^{-1}(r)$ is a region of A with the same signature as r (see Def. 1.76 for the definition of the simulation map η_A). Therefore, the property of event effectiveness is satisfied in A. From any place $p \in {}^\bullet e \setminus {}^\bullet e'$ (respectively $p \in e^\bullet \setminus e'^\bullet$), one can construct similarly a region r' of A such that $r' \in {}^\circ e \setminus {}^\circ e'$ (resp. $r \in e^\circ \setminus e'^\circ$). Therefore, the property of event simpleness is satisfied in A. Conversely, assume that A is an initialized transition system and that A satisfies the properties of event-state separation, event-effectiveness and event-simpleness. Then the synthesized net system $N = SN(A)$ is an elementary net system (by event effectiveness and event simpleness) and $A \rhd RG(N)$ (by Prop. 1.75). □

Corollary 1.82. *A minimal initialized transition system $A \cong Min(A)$ is an elementary transition system if and only if it satisfies the properties of event-state separation, event-effectiveness and event-simpleness.* □

1.7 Net Synthesis from Languages

The net synthesis problem from initialized transition systems A up to language equivalence consists of deciding whether $L(A) = L(N)$ for some net system N and constructing such an N if it exists. This problem has received little attention in the literature on elementary (or quasi-elementary) nets. We approach the problem in two steps. In a first step, we extend the theory of regions to arbitrary languages $L \subseteq E^*$, identified with infinite trees, i.e., with infinite initialized transition systems without converging paths. In a second step, we focus on languages $L(A)$ of finite initialized transition systems, and we adapt the results to finite unfoldings $\mathcal{U}(A)$ of the given transition systems A, located between A and $L(A)$, i.e., $L(A) \rhd \mathcal{U}(A) \rhd A$.

In Sec. 1.6, we have illustrated with Example 1.78 a situation in which an initialized transition system $A = Min(RG(N))$ cannot be realized by any net system, although N is an elementary net system which realizes A up to language equivalence. Further, we argued that the trouble comes from the loss of some signatures of regions through the minimization operation $RG(N) \rhd Min(RG(N))$. In order to realize an initialized transition system A by a net system N up to language equivalence, if this is possible, one may be forced to synthesize the places of N from the regions of some *unfolding* of A. In Example 1.78, A is an *acyclic* transition system, i.e., $\delta(s,u) = s \Rightarrow u = \varepsilon$ for all states s and for all sequences of events u, and the choice of the right unfolding is not a problem: there exists only one unfolding, that splits s_6 to two different states, according to the label d or c of the incoming path, and thus produces an initialized transition system A' such that $A' \cong RG(N)$ and $A' \rhd A$. In the general case where A may have cycles, the choice of the right unfolding is a bit more complex. To help the presentation, we make a first attempt by considering complete and hence possibly infinite unfoldings of initialized transition systems.

Definition 1.83. *A language $L \subseteq E^*$ is* prefix closed *if $uv \in L$ entails $u \in L$ for all $u, v \in E^*$ (in particular, L contains the empty word ε). For any prefix closed language $L \subseteq E^*$, let $L = (S, E, \delta, s_0)$ be the initialized transition system defined by $S = L$, $s_0 = \varepsilon$, and $\delta(u,e) = ue$ if and only if $u \cdot e \in L$ (language L is both the set of states and the set of labels of paths of this transition system, e.g., Fig 1.27). For any initialized transition system $A = (S, E, \delta, s_0)$, let $\equiv_A$ be the equivalence relation on the language $L(A)$ given by $u_1 \equiv_A u_2 \;\Leftrightarrow\; \delta(s_0, u_1) = \delta(s_0, u_2)$ where $\delta : S \times E^* \to S$ is the inductive extension of $\delta : S \times E \to S$ (see Def. 1.29 on page 22).* ◇

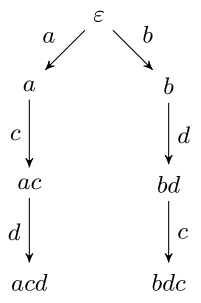

$$\begin{tikzcd}
 & \varepsilon \arrow[dl, "a"'] \arrow[dr, "b"] & \\
a \arrow[d, "c"'] & & b \arrow[d, "d"] \\
ac \arrow[d, "d"'] & & bd \arrow[d, "c"] \\
acd & & bdc
\end{tikzcd}$$

Fig. 1.27. the initialized transition system version of a language

Example 1.84. Let A be the initialized transition system shown on the right of Fig. 1.26. The initialized transition system L defined from $L(A)$ is shown in Fig. 1.27. The equivalence $\equiv_A$ on $L(A)$ identifies acd and bdc ($acd \equiv_A bdc$). A is isomorphic to the quotient of L by the equivalence $\equiv_A$. □

The following lemma states that folding and unfolding operations may be characterized in terms of the languages $L(A)$ and the quivalence relation $\equiv_A$.

Lemma 1.85. *For any initialized transition systems A and A', $A \rhd A' \Leftrightarrow (L(A) = L(A') \wedge \equiv_A \subseteq \equiv_{A'})$. In particular, for any initialized transition system A' with set of events E and for any prefix closed language $L \subseteq E^*$, $L \rhd A' \Leftrightarrow L = L(A')$.*

Proof. The left-to-right implication follows from the definition of saturating morphisms by straightforward induction on the length of words. In order to establish the converse implication, assume $L(A) = L(A')$ and $\equiv_A \subseteq \equiv_{A'}$. Let S and δ (resp. S' and δ') be the set of states and partial transition function of A (resp. A'). Let $\varphi : S \to S'$ be the map defined by $\varphi(s) = \delta'(s'_0, u)$ for any u such that $\delta(s_0, u) = s$. As $L(A) = L(A')$ and $\equiv_A \subseteq \equiv_{A'}$, φ is well defined and total, and it is a label preserving morphism of initialized transition systems. As $L(A') \subseteq L(A)$, φ is a saturating morphism, hence $A \rhd A'$. The main statement of the lemma has been established. The particular case follows by identifying L with the initialized transition system A defined from L (see Def. 1.83) and by remarking that in this case, $\equiv_A$ is the identity (on the language L). □

From now on in the section, we shall often use the same notation to denote a prefix-closed language L and the initialized transition system defined from this language according to Def. 1.83. At this stage, it is important to note that the definition of regions (Def. 1.20) does not rely on the assumption that transition systems are finite. Therefore, the notion of regions may be extended to arbitrary prefix-closed languages over a finite set of events E.

Definition 1.86. *Given a prefix-closed language $L \subseteq E^*$, a region of L is a region of the initialized transition system $L = (L, E, \delta, \varepsilon)$ with the partial transition map $\delta(u, e) = ue$ if $ue \in L$.* ♢

Remark 1.87. A signature $r : \{init\} \cup E \to \{-1, 0, 1\}$ determines a region of L if and only if there exists a corresponding map $r : L \to \{0, 1\}$ (necessarily unique) such that $r(\varepsilon) = r(init)$ and $r(ue) = r(u) + r(e)$ for all words $ue \in L$ with $e \in E$. As a consequence, a region of L restricts on any prefix-closed language $L' \subseteq L$ to a region of L' with the same signature (assuming that each event $e \in E$ occurs at least once in both L and L'). □

For any language $L \subseteq E^*$ over a finite alphabet E, the set $R(L)$ of all regions of L is finite, because there exists only a finite number of signatures of regions. Therefore, for any $L \subseteq E^*$, $SN(L)$ is a well-defined net system. It follows from this observation that most results established in the basic theory of regions may be applied to languages with minor adaptations. The following proposition is a joint adaptation of Th. 1.37 and Property 1.62 following from Th. 1.51.

Proposition 1.88. *Let L be a prefix closed language over a finite set of events E. Then $L = L(N)$ for some net system N with the set of transitions E if and only if $L = L(N)$ for $N = SN(L)$, if and only if the event-state separation property ESSP holds in the initialized transition system L. If ESSP does not hold, then the language $L(N)$ of the net system $N = SN(L)$ synthesized from all regions of L is the least language of a net system larger than L.*

Proof. Let N be a net system. By Lemma 1.85, $L = L(N)$ if and only if $L \rhd RG(N)$. By Prop. 1.75, $L \rhd RG(N)$ if and only if $N \cong SN_R(L)$ for some set of regions $R \subseteq R(L)$ such that $ESSP(R)$ holds in L. If $ESSP(R)$ holds in L for $R \subseteq R(L)$, then it holds also for $R = R(L)$. By Prop. 1.75, ESSP holds in L if and only if $L \rhd RG(SN(L))$, if and only if $L = L(N)$ for $N = SN(L)$. Therefore, $L = L(N)$ for some net system N if and only if the event-state separation property ESSP holds in L, if and only if $L = L(N)$ for $N = SN(L)$. The first statement of the proposition has been established. In order to establish the second statement, observe that for any prefix-closed language L and for any initialized transition system A, $L \leq A$ if and only if $L \subseteq L(A)$. So, because $L \leq RG(SN(L))$ (by Property 1.60 following from Th. 1.51), L is included in the language of the net system $SN(L)$ synthesized from all regions of L. For any other net system N such that $L \subsetneq L(N)$, every place p of N determines a region of $L(N)$ with the same signature. Since $L \leq L(N)$, by Lemma 1.56, this signature coincides with the signature of some region of L, and therefore with the signature of some place of the net system $SN(L)$. The above reasoning shows that N is isomorphic to a restriction of $SN(L)$, hence $L(N)$ necessarily contains the language of the net system $SN(L)$. □

Remark 1.89. Prop. 1.88 has theoretical interest, but it does not help much in practice since it does not tell us how computing $SN(L)$, even if $L = L(A)$ is the language of a finite initialized transition system A. In the end of the section, we show that in that case, the optimal net system $SN(L)$ may be computed effectively. Let $L = L(A)$ be the language of a finite initialized transition system $A = (S, E, \delta, s_0)$. The map from L to S that sends each word of L to the state of A in which it is recognized is a saturating morphism from L (seen as an initialized transition system) to A, hence $L \rhd A$ and in particular $L \leq A$. Therefore, every region of A induces a region of L with the same signature (by Lemma 1.56). However, as was shown by Example 1.78, not every region of L corresponds in this way to a region of A. Therefore, the language of the net system $SN(L)$ synthesized from all regions of L may be *smaller* than the language of the net system $SN(A)$ synthesized from all regions of A. The next definition proposes a *finite* unfolding $\mathcal{U}(A)$ of A ($L(A) \rhd \mathcal{U}(A) \rhd A$) such that

$L(A)$ and $\mathcal{U}(A)$ have the same signatures of regions. In view of Prop. 1.88, the language of the net system $SN(\mathcal{U}(A))$ synthesized from $\mathcal{U}(A)$ is therefore the least language of a net system larger than or equal to $L(A)$.

Definition 1.90. *The* limited unfolding *of a finite initialized transition system $A = (S, E, \delta, s_0)$ with language $L = L(A)$ is the initialized transition system $\mathcal{U}(A) = (S', E, \delta', s'_0)$ defined as follows:*

1. *S' is the set of words $u \in L$ such that, for any decomposition $u = u_1u_2u_3$ with $u_2 \neq \varepsilon$, $\delta(s_0, u_1) \neq \delta(s_0, u_1u_2)$ in A,*
2. *the initial state s'_0 is the empty word ε,*
3. *$\delta'(u, e)$ is defined if and only if $\delta(s_0, ue)$ is defined in A, and then, either $\delta'(u, e) = ue$ if $ue \in S'$, or $\delta'(u, e) = u_1$ for the (unique) prefix u_1 of u $(= u_1u_2)$ such that $\delta(s_0, ue) = \delta(s_0, u_1)$.* ♢

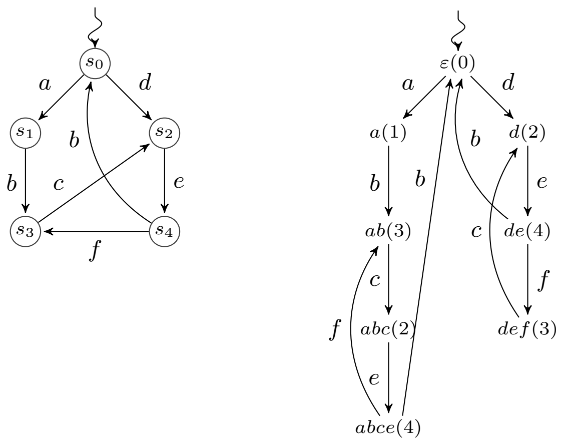


Fig. 1.28. A and its limited unfolding $\mathcal{U}(A)$

Example 1.91. Fig. 1.28 shows an initialized transition system A and its limited unfolding $\mathcal{U}(A)$. The word *abce* is a state of $\mathcal{U}(A)$ because no state of A is visited twice when following the corresponding path from the initial state 0 of A. On the contrary, the word *abcef* is not a state of $\mathcal{U}(A)$ because *abcef* leads to the same state of A as the word *ab* which is a prefix of *abcef*. $\mathcal{U}(A)$ folds back to A as indicated by the labels attached to the states of $\mathcal{U}(A)$, e.g., the state *abce* is mapped to the state 4 of A and the state *ab* is mapped to the state 3 of A. □

Note that the set S' in Def. 1.90 is necessarily finite because the maximal length of the words $u \in S'$ is at most $|S|$ (the number of states of A) and E is a finite set of events. The transitions of $\mathcal{U}(A)$ may be classified in two categories: *forward* transitions $u \xrightarrow{e} ue$ and *backward* transitions $uv \xrightarrow{e} u$. By cutting off the backward transitions, one gets a finite tree that spans $\mathcal{U}(A)$. The backward transitions are chords of this tree, retracing the incoming paths.

Remark 1.92. The relation $\mathcal{U}(A) \rhd A$ is justified by the (saturating) simulation map that sends any word $u \in S'$ to $\delta(s_0, u)$. The relation $L \rhd \mathcal{U}(A)$ follows from $L(\mathcal{U}(A)) = L(A)$; it is justified by the (saturating) simulation map φ defined inductively by $\varphi(u) = u$ if $u \in S'$ and $\varphi(u_1u_2u_3) = \varphi(u_1u_3)$ if $\delta(s_0, u_1u_2) = \delta(s_0, u_1)$ (Exer. 1.6). □

Note that $\mathcal{U}(A)$ may have size exponential in the size of A. However, constructing this limited unfolding in order to obtain indirectly a net realization of the language of A is justified by the following proposition, which states the main result of the section.

Proposition 1.93. *Let A be a finite initialized transition system with set of events E, and let $r : \{init\} \cup E \to \{-1, 0, 1\}$ be the signature of a region of the language $L(A)$. Then there exists a region of $\mathcal{U}(A)$ with the same signature r.*

Proof. Let $A = (S, E, \delta, s_0)$, $L = L(A)$, and $\mathcal{U}(A) = (S', E, \delta', s'_0)$. As $r : \{init\} \cup E \to \{-1, 0, 1\}$ is the signature of a region of L, this map determines a (unique) map $r : L \to \{0, 1\}$ such that $r(ue) = r(u) + r(e)$ for all $ue \in L$ with $e \in E$. We claim that the induced restriction $r' : S' \to \{0, 1\}$ of r on $S' \subseteq L$ is a region of $\mathcal{U}(A)$, with the given signature $r : \{init\} \cup E \to \{-1, 0, 1\}$. In order to establish this claim, in view of Remark 1.52, it suffices to prove that $\delta'(u, e) = v$ entails $r(v) = r(u) + r(e)$ for any word $u \in S'$ and for any event $e \in E$. If $v = ue$, then $u \xrightarrow{e} ue$ is a forward transition of $\mathcal{U}(A)$, and $r(v) = r(u) + r(e)$ follows since r is a region of L. In the opposite case, $u \xrightarrow{e} v$ is a backward transition of $\mathcal{U}(A)$, i.e., $u = vw$ for some $w \in E^*$ with $\delta(s_0, v) = \delta(s_0, vwe)$. Let $w = e_1e_2 \dots e_m$ (with $m = 0$ if $w = \varepsilon$) and let $z = r(e_1) + r(e_2) \dots + r(e_m) + r(e)$. Then, $r(v) = r(u) + r(e)$ if and only if $r(v) = r(v) + z$ if and only if $z = 0$. Now $vwewe \in L$ because $\delta(s_0, v) = \delta(s_0, vwe)$. As $r(vwe) = r(v) + z$ and $r(vwewe) = r(v) + 2z$, and because $r(v)$ and $r(vwewe)$ belong to $\{0, 1\}$, necessarily $z = 0$. Therefore, $r(v) = r(u) + r(e)$ for every backward transition $u \xrightarrow{e} v$ of $\mathcal{U}(A)$. □

By Lemma 1.56, every signature of a region of $L(A)$ coincides with the signature of a region of $\mathcal{U}(A)$. By Prop. 1.93, every signature of a region of $\mathcal{U}(A)$ coincides with the signature of a region of $L(A)$. Therefore, for any finite initialized transition system A, the signatures of the regions of the language $L(A)$ are the same as the signatures of the regions of the limited unfolding $\mathcal{U}(A)$ of A. The theorem below follows as an immediate corollary.

Theorem 1.94. *The net system $SN(\mathcal{U}(A))$ synthesized from the limited unfolding of A recognizes the least language of a net system larger than or equal to $L(A)$.*

Example 1.95 (Exple. 1.27 (page 22) continued). Fig. 1.29 shows (on the left) the limited unfolding $A' = \mathcal{U}(A)$ of the initialized transition system A given on the left of Fig.1.15 (on page 22), the net system $SN(\mathcal{U}(A))$ (in the center) synthesized from it, and the reachability graph of this net (on the right). The latter differs from the reachability graph of $SN(A)$, which was shown in Fig. 1.15. Both reachability graphs generate the same language, which is larger than the language of A. □

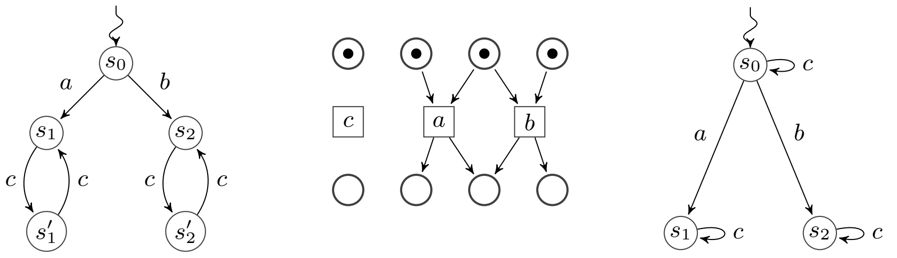


Fig. 1.29. A', $SN(A')$, and $RG(SN(A'))$ where $A' = \mathcal{U}(A)$ is the limited unfolding of initialized transition system A from Fig.1.15 on page 22

Example 1.96 (Exple. 1.78 on page 44 continued). Let A be the initialized transition system depicted in the rightmost part of Fig. 1.26 on page 45. $\mathcal{U}(A)$ is the initialized transition system depicted in Fig. 1.27 (on page 47). The reachability graph of the net $SN(\mathcal{U}(A))$ is isomorphic to $\mathcal{U}(A)$ (see Fig. 1.26). The synthesized net system $SN(\mathcal{U}(A))$ realizes here exactly the language $L(A)$. □

The next proposition shows that limited unfoldings can actually be used to decide on the exact net realization problem for languages of finite initialized transition systems.

Proposition 1.97. *Let $L = L(A)$ be the language of a finite initialized transition system A.*

1. *If the property $ESSP(R)$ holds in L for some set of regions $R \subseteq R(L)$, then $ESSP(R')$ holds in $\mathcal{U}(A)$ for R' defined as the set of regions of $\mathcal{U}(A)$ with signatures equal to signatures of regions in R.*
2. *If the property $ESSP(R')$ holds in $\mathcal{U}(A)$ for some set of regions $R' \subseteq R(\mathcal{U}(A))$, then $ESSP(R)$ holds in L for $R = \{\varphi^{-1}(r') \mid r' \in R'\}$, where φ is the simulation map which justifies the relation $L \rhd \mathcal{U}(A)$ (see Remark 1.92).*

Proof. Let $\mathcal{U}(A) = (S', E, \delta', s'_0)$, thus $S' \subseteq L$ by Def. 1.90. Suppose $u \in S' \subseteq L$ and $ue \notin L$ for some event e. If $ESSP(R)$ holds in L, then some region $r \in R(L)$ separates e from u in the transition system L. The induced restriction of r on $S' \subseteq L$ is a region r' of $\mathcal{U}(A)$ with the same signature as r, hence it separates e from u in $\mathcal{U}(A)$. This establishes the first statement. In order to prove the second statement, let $u \in L$ and $ue \notin L$, and suppose that $ESSP(R')$ holds in $\mathcal{U}(A)$. Let $v = \varphi(u)$, where $\varphi : L \to \mathcal{U}(A)$ is the simulation morphism defined in Remark 1.92, then $\delta(s_0, u) = \delta(s_0, v)$, hence $ve \notin L$. As $L \rhd \mathcal{U}(A)$ entails $L(\mathcal{U}(A)) = L$, $\delta'(v, e)$ is undefined in $\mathcal{U}(A)$, hence some region $r' \in R'$ separates e from v. By Lemma 1.56, $\varphi^{-1}(r')$ is a region in $R(L)$ with the same signature as r', hence separating e from u in L. □

Corollary 1.98. *The language $L(A)$ of a finite initialized transition system A may be realized by some net system if and only if $ESSP(R)$ holds in $\mathcal{U}(A)$ for some set of regions R, and in this case, $L(A) = L(SN_R(\mathcal{U}(A)))$.* □

In view of Prop. 1.81, the above corollary may be strengthened into the following.

Corollary 1.99. *The language $L(A)$ of a finite initialized transition system A may be realized by some* elementary *net system if and only if the properties of event-state separation, event-effectiveness, and event-simpleness are satisfied in $\mathcal{U}(A)$, and in this case, $L(A) = L(SN_R(\mathcal{U}(A)))$ for any set of regions R of $\mathcal{U}(A)$ witnessing for these properties.* □

1.8 Regions of Labelled Partial 2-Structures

The scope of Ehrenfeucht and Rozenberg's theory of regions extends beyond Elementary Net Synthesis. The primary goal of this theory was to provide an effective representation of Labelled Partial 2-Structures by Labelled Partial Set 2-Structures, that encompass reachability graphs of Elementary Nets as a particular case. In this section, we outline the theory of regions of Labelled Partial 2-Structures which was defined in [21, 22] and the articulation between Elementary Net Synthesis and this more general theory.

Labelled Partial 2-Structures are a subclass of labelled graphs that may be defined as follows (the definition below is equivalent to the definition given in [21]).

Definition 1.100. *A* Labelled Partial 2-Structure *(or LP2S) is a tuple $g = (S, F, E, \lambda)$ where $S = dom(g)$ is a set of* states *(or nodes), called the* domain *of g, $F \subseteq \{(x, y) \mid x, y \in S \wedge x \neq y\}$ is a set of* 2-edges*, $E = alph(g)$ is an* alphabet*, and $\lambda : F \to E$ is a* labelling map*. By extension, let $g = (S, F, \sim, E, \lambda)$ where $\sim$ is the equivalence relation on F induced by the labelling map λ, i.e., $(x, y) \sim (x', y')$ if and only if $\lambda(x, y) = \lambda(x', y')$.* ♢

According to this definition, a LP2S is essentially a loop-free and simple transition system (see Defs. 1.29 and 1.30) that is not initialized and needs not be connected. The theory of regions was founded in the perspective of characterizing those LP2S which may be represented up to an isomorphism using sets and ordered symmetric differences of sets for representing nodes and edge-labels, respectively. Such representations are called Labelled Partial Set 2-Structures.

Definition 1.101. *A* Labelled Partial Set 2-Structure *(or LPS2S)* based on *a (non-empty) set B is an LP2S $g = (S, F, \sim, E, \lambda)$ where $S \subseteq 2^B$, $E \subseteq \{(x,y) \mid x, y \subseteq B \wedge x \cup y \neq \emptyset \wedge x \cap y = \emptyset\}$, and for every 2-edge $(x,y) \in F$, $\lambda(x,y) = (x \setminus y, y \setminus x)$ is the* ordered symmetric difference *between the sets x and y (also noted $osd(x,y)$).* ♢

In order to make the representation problem for LP2S quite precise, it remains to specify their isomorphisms.

Definition 1.102. *Let $g_1 = (S_1, F_1, \sim_1, E_1, \lambda_1)$ and $g_2 = (S_2, F_2, \sim_2, E_2, \lambda_2)$ be LP2S. A* morphism *of LP2S $\varphi : g_1 \to g_2$ is a map $\varphi : S_1 \to S_2$ such that,*

1. $\forall (x,y) \in F_1 \qquad \varphi(x) = \varphi(y) \quad or \quad (\varphi(u), \varphi(v)) \in F_2$
2. $\forall (x,y),\ (u,v) \in F_1 \quad s.t. \quad \varphi(x) \neq \varphi(y) \ \ and \ \ \varphi(u) \neq \varphi(v)$

$$(x,y) \sim_1 (u,v) \quad \Rightarrow \quad (\varphi(x), \varphi(y)) \sim_2 (\varphi(u), \varphi(v))$$

A morphism $\varphi : g_1 \to g_2$ is said to be uniform *if, for all $(x,y), (u,v) \in F_1$, $(x,y) \sim_1 (u,v)$ and $\varphi(x) \neq \varphi(y)$ entail $\varphi(u) \neq \varphi(v)$. A morphism $\varphi : g_1 \to g_2$ is said to be* strongly surjective *if the map $\varphi : S_1 \to S_2$ is onto and every 2-edge in F_2 is the image $(\varphi(x), \varphi(y))$ of some 2-edge $(x,y) \in F_1$. A morphism $\varphi : g_1 \to g_2$ is an* isomorphism *if the map $\varphi : S_1 \to S_2$ is bijective and the inverse map $\varphi^{-1} : S_2 \to S_1$ defines a morphism from g_2 to g_1.* ♢

The representation problem for LP2S is the following: given a LP2S g, decide whether exists and construct a LPS2S h such that g and h are isomorphic as LP2S. This problem was solved in [21]. The crucial idea is that, if a LP2S g is isomorphic to some LPS2S, then it must be isomorphic to some LPS2S based on *regions* of g defined as follows.

Definition 1.103. *Let $g = (S, F, \sim, E, \lambda)$ be a LP2S. A subset $r \subseteq S$ is a* region *of g iff, for all $(x,y), (u,v) \in F$, $(x,y) \sim (u,v)$ entails the following:*

1. $(x \in r \ \wedge \ y \notin r) \ \Rightarrow \ (u \in r \ \wedge \ v \notin r)$
2. $(x \notin r \ \wedge \ y \in r) \ \Rightarrow \ (u \notin r \wedge v \in r)$

Let R_g denote the set of regions of g, and for every $x \in S$, let

$$R_g(x) \ = \ \{r \in R_g \mid x \in r\}$$ ♢

Def. 1.20 in Sect. 1.3 is a mere restatement of this original definition. The general representation problem for LP2S differs to some extent from the net realization problem for initialized transition systems, which was dealt with in Sections 1.3 and 1.4, since the former problem does not depend at all upon any firing rule. In spite of this difference, the construction, from a given LP2S g, of a LPS2S $regv(g)$, called the regional version of g, has much in common with the direct construction, from an initialized and separated transition system A, of the reachability graph $RG(SN(A))$ of the net system $SN(A)$ synthesized from A.

Definition 1.104. *Given a LP2S $g = (S, F, \sim, E, \lambda)$, let $reg_g : S \to 2^{R_g}$ be the (*regional*) mapping defined by $reg_g(x) = R_g(x)$.* ♢

Definition 1.105. *Given a LP2S $g = (S, F, \sim, E, \lambda)$, the* regional version *of g is the LPS2S $regv(g) = (S', F', \sim', E', \lambda')$, based on the set $R_g \setminus \emptyset$, defined by:*
1. $S' = \{R_g(x) \mid x \in S\}$,
2. $F' = \{(R_g(x), R_g(y)) \mid x, y \in S \wedge (x, y) \in F \wedge R_g(x) \neq R_g(y)\}$,
3. $E' = \{osd(x', y') \mid (x', y') \in F'\}$, and
4. $\lambda'(x', y') = osd(x', y')$. ♢

It was shown in [21] that for any LP2S g, reg_g is a uniform and strongly surjective morphism from g onto $regv(g)$. This morphism is bijective if and only if g satisfies the following Node Separation Property (NS), which was called State Separation in Sect. 1.4:

$$(\forall x, y \in dom(g)) \quad x \neq y \Rightarrow R_g(x) \neq R_g(y).$$

The solution to the representation problem for LP2S is given by the theorem stated below, proven in [21], where ES denotes the Event Separation Property defined as follows for $g = (S, F, \sim, E, \lambda)$:

$$\forall (x, y), (u, v) \in F \quad \lambda(x, y) \neq \lambda(u, v) \Rightarrow (R_g(x) \setminus R_g(y)) \neq (R_g(u) \setminus R_g(v))$$

which may be restated (using complementary regions) equivalently as:

$$\lambda(x, y) \neq \lambda(u, v) \Rightarrow osd(R_g(x), R_g(y)) \neq osd(R_g(u), R_g(v)).$$

Theorem 1.106. *For any LP2S g, the following are equivalent:*
1. g is isomorphic to some LPS2S h,
2. g is isomorphic to $regv(g)$,
3. reg_g is an isomorphism,
4. g has the properties NS and ES.

Based on this theorem, two decision procedures were proposed in [21] for the LP2S representation problem. One procedure consists of computing the regional version $regv(g)$ of the given LP2S g and then checking that g and $regv(g)$ are isomorphic. The other procedure consists of checking g for the separation properties NS and ES and then computing $regv(g)$ only if both properties hold. One finds here an anticipation of the two types of decision procedures for the net realization problem of initialized transition systems which were presented in Sections 1.3 and 1.4. It was moreover shown in [21] that for any LP2S g satisfying NS and ES, the LPS2S $regv(g)$ is *maximal* in the sense that any LPS2S h isomorphic to g derives from $regv(g)$ by a bijective renaming of the base, possibly followed by the addition to the base of useless elements b, i.e., elements b such that $\forall x, y \in dom(h)\ \ b \in x \Leftrightarrow b \in y$. This property of maximality is an early form of the property of saturatedness of synthesized net systems observed in Sect.1.5 (we recall that a net system is saturated if no place can be added to this net system without modifying its reachability graph up to isomorphisms of initialized transition systems).

The approach briefly recalled above was adapted in [22] to provide an effective solution to the net synthesis problem from initialized transition systems. This application of the general theory of regions of Labelled Partial 2-Structures to concurrency relies on two central facts reported herafter.

First, given any LP2S g, one can extract from $regv(g)$ a net $N(g) = (P, E, F)$ defined as follows. The set of places P is the base of $regv(g)$, i.e. the set R_g of all regions of g. The set of transitions E is the alphabet of $regv(g)$, i.e. the set of ordered symmetric differences $osd(x, y)$ for 2-edges (x, y) of $regv(g)$. The flow relation $F \subseteq R_g \times T \cup T \times R_g$ is defined as $(r, (x, y)) \in F$ iff $r \in x$ and $((x, y), r) \in F$ iff $r \in y$.

Second, given any LPS2S $h = (S, F, \sim, E, \lambda)$, where $S \subseteq 2^B$ and $E \subset 2^B \times 2^B$ letting B be the base of h, let FC (Forward Closure) denote the property defined as follows:

$$\begin{gathered}(\forall (A, A') \in E)\ \ (\forall x \in S)\ \ A \subseteq x\ \wedge\ A' \cap x = \emptyset\ \Rightarrow \\ (\exists y \in S)\ \ (x, y) \in F\ \wedge\ \lambda(x, y) = (A, A').\end{gathered}$$

Then FC is a *canonical* property in the following sense: if a LP2S g is isomorphic to an LPS2S, then g is isomorphic to some LPS2S h satisfying FC if and only if $regv(g)$ satisfies FC. Note that FC is essentially the same as the Event-State Separation Property considered in Section 1.4. In the sequel, a LPS2S which satisfies FC is said to be *forward closed*.

Theorems 1.37 and 1.43 of this book are inspired from similar results, based on the above two facts, established in [22] for Initialized Labelled Partial 2-Structures (ILP2S) and Initialized Labelled Partial Set 2-Structures (ILPS2S). ILP2S and ILPS2S are just LP2S and LPS2S with initial nodes. The Elementary Net Systems studied in [22] are a bit different from those we have considered: every place belongs to the pre-set or post-set of some transition, and every transition has pre-places *and* post-places. It was shown in [22] that an ILP2S (g, s_0) is isomorphic to the reachability graph of some Elementary Net System N iff (g, s_0) is reachable, g is isomorphic to $regv(g)$, and $regv(g)$ satisfies FC. Equivalently, (g, s_0) is isomorphic to the reachability graph of some Elementary Net System N iff (g, s_0) is reachable, g satisfies NS and ES, and $regv(g)$ satisfies FC. If these conditions hold, then (g, s_0) is isomorphic to the reachability graph of the saturated net system $NS(g)$ defined as $N(regv(g))$ with $R_g(s_0)$ as the initial marking.

The synthesis problem was also solved in [22] for C/E Net Systems, which differ from Elementary Net Systems only in that transitions may be fired forwards or backwards. In order to characterize ILP2S that may be realized by C/E Net Systems, it suffices to require, in addition to Forward Closure (FC), a similar property of Backward Closure (BC). Ehrenfeucht and Rozenberg asked in the conclusion of [22] the following question, that will be answered in Part 2 of this book: *Is there a notion of a "morphism" between labelled 2-structures that would give a characterization of regions in the same way that c-morphisms between 2-structures characterize clans of 2-structures?.*

In the end of the section, we sketch a different extension of Ehrenfeucht and Rozenberg's theory of regions, aiming at the representation of an LP2S or ILP2S by a C/E or Elementary Net determined entirely by its reachability set, i.e., the transitions may be left implicit and they can be reconstructed unambiguously from the considered set of markings.

From now on, non-uniform morphisms of LP2S are ignored, i.e., all morphisms of LP2S are uniform (see Def. 1.102). Let us introduce some definitions.

Definition 1.107. *Given a (non-empty) set B, let $2^B = (2^B, F, \sim, 2^B \times 2^B, osd)$ denote the free LPS2S over the base B, i.e. the LPS2S with the set of 2-edges $F = \{(x, y) \mid x, y \subseteq B \wedge x \neq y\}$.* $\diamondsuit$

Definition 1.108. *Let $g_1 = (S_1, F_1, \sim_1, E_1, \lambda_1)$ and $g_2 = (S_2, F_2, \sim_2, E_2, \lambda_2)$ be LP2S. Then g_1 is a* substructure *of g_2 if $S_1 \subseteq S_2$, $F_1 \subseteq F_2$, $\sim_1 = \sim_2 \cap (F_1 \times F_1)$, $E_1 \subseteq E_2$, and λ_1 is jointly the restriction of λ_2 on F_1 and the co-restriction of λ_2 on E_1. If moreover $F_1 = F_2 \cap (S_1 \times S_1)$, then g_1 is a* full substructure *of g_2.* $\diamondsuit$

The main contribution of [7] is the following theorem.

Theorem 1.109. *Given a LP2S g, there exists a forward-closed and full substructure $\overline{g}$ of 2^{R_g} and a morphism $\overline{reg_g} : g \to \overline{g}$ such that, for any base set B and for any forward-closed and full substructure h of 2^B, every morphism $\varphi : g \to h$ factors uniquely as $\varphi = \psi \circ \overline{reg_g}$, where $\psi : \overline{g} \to h$.* $\square$

We shall not give the proof of this theorem but just sketch the construction of $\overline{g}$. The idea is to construct the LPS2S $\overline{g}$ as the limit of an increasing sequence of substructures g_n of 2^{R_g} defined inductively from $g_0 = regv(g)$, where increasing means that for every n, g_n is a substructure of g_{n+1}. This increasing sequence must stabilize since 2^{R_g} is finite. For each n, g_{n+1} is obtained from g_n in two expansion steps. In the first step, one takes the set of nodes $S_n \subseteq 2^{R_g}$ of g_n and one constructs the full substructure of 2^{R_g} over the considered set of nodes. In the second step, for every x, y, z in S_n such that $x \setminus y \subseteq z$ and $(y \setminus x) \cap z = \emptyset$, one adds a 2-edge (z, w) from z to $w = (z \setminus (x \setminus y)) \cup (y \setminus x)$, if such edge is not already present. Let $\overline{S}$ be the set of nodes of the fixpoint $g_n = g_{n+1}$. Then the regions of $\overline{g}$ are exactly the subsets of $\overline{S}$ of the form $\{x \in \overline{S} \mid r \in x\}$ for some $r \in R_g$. As a consequence, the sets of regions R_g and $R_{\overline{g}}$ are isomorphic when seen as partial Boolean algebras (where $x \vee y$ is defined and equal to $x \cup y$ if $x \cup y$ is a region, and it is undefined otherwise). In other words, regions are robust: expanding $regv(g)$ (or any LPS2S) by adding iteratively new transitions between existing nodes and new nodes to ensure forward closedness w.r.t. existing transitions, does not affect the structure of regions!

The following theorem follows as an easy corollary from Th. 1.109.

Theorem 1.110. *A LP2S g is isomorphic to a forward-closed and full substructure of a LPS2S iff it is isomorphic to $\overline{g}$, iff $\overline{reg_g}$ is an isomorphism, iff the underlying graph of g is complete and all three axioms NS, ES, and FC are satisfied in g.* □

Further readings

An intriguing connection between Elementary Net Synthesis and algebra was pointed to in [6]. There, it was observed that the set of regions of an initialized transition system, or more generally of a labelled partial 2-structure, forms a cohererent orthomodular poset, a structure which has been studied at depth in the framework of models of quantum logics. Coherent orthomodular posets may alternatively be seen as transitive partial Boolean algebras.

In [6], a dual adjunction based on regions is constructed between Prime Coherent Orthomodular Posets (PCOP) and C/E Transition Systems (CETS), i.e., transition systems isomorphic to C/E net reachability graphs. The dual adjunction is comprised of two contravariant functors $H : CETS^{op} \to PCOP$ and $J : PCOP^{op} \to CETS$, such that H^{op} is left adjoint to J (equivalently, J^{op} is left adjoint to H). This situation is conceptually close to the situation found in Sec. 1.5, where we established an order-theoretic Galois connection between Net Systems and Initialized Transition Systems. However, it is still a conjecture whether $H(A) \cong H(J(H(A)))$ and $J(P) \cong J(H(J(P)))$ for every CETS A and for every PCOP P, as should be the case for a categorical Galois connection. Answering positively this conjecture would show that regions are indeed much more robust than was shown in [7].

Problems

1.1 (From [20]). Let $R \subset R(A)$ be an admissible subset of regions of an initialized transition system A. A region $r \in R$ is redundant in R if $R \setminus \{r\}$ is admissible. Show that r is redundant in R in each of the following cases:
(a) $S \setminus r \in R$,
(b) $r = r_1 \cap r_2$ and $S \setminus r = r_3 \cup r_4$ for some r_1, r_2, r_3, and r_4 in R,
(c) $r = r_1 \cup r_2$ and $S \setminus r = r_3 \cap r_4$ for some r_1, r_2, r_3, and r_4 in R,
(d) $r = r_1 \cap r_2$ for some r_1 and r_2 in R such that

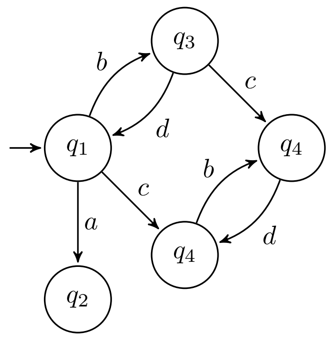
$$\forall s \in S \quad \forall e \in E \quad \forall s' \in S \setminus r \qquad s \xrightarrow{e} s' \Rightarrow s' \notin r_1 \cup r_2$$

1.2 (From [5]). A minimal region of an initialized transition system A is a region of A which is minimal for set inclusion amongst the non-trivial regions of A. Show successively the following:
(a) The union of two disjoint regions r_1 and r_2 is a region with

$$
\begin{aligned}
{}^\bullet(r_1 \cup r_2) &= ({}^\bullet r_1 \cup {}^\bullet r_2) \setminus (({}^\bullet r_1 \cap r_2{}^\bullet) \cup ({}^\bullet r_2 \cap r_1{}^\bullet))\\
(r_1 \cup r_2)^\bullet &= (r_1{}^\bullet \cup r_2{}^\bullet) \setminus (({}^\bullet r_1 \cap r_2{}^\bullet) \cup ({}^\bullet r_2 \cap r_1{}^\bullet))
\end{aligned}
$$

(b) The set-theoretical difference $r_2 \setminus r_1$ of two regions r_1 and r_2 such that $r_1 \subseteq r_2$ is a region.
(c) Every region is a disjoint union of minimal regions. Find an example showing that this decomposition is not necessarily unique.
(d) If r is a region and $e \in {}^\bullet r$, then there exists a minimal region $r_1 \subseteq r$ such that $e \in {}^\bullet r_1$. Symmetrically, if r is a region and $e \in r^\bullet$, then there exists a minimal region $r_2 \subseteq r$ such that $e \in r_2{}^\bullet$.
(e) The set of minimal regions of an elementary transition system is an admissible set of regions.

1.3 (From [20]).
(a) Compute all non-trivial regions of the initialized transition system displayed next.
(b) Construct the elementary net system synthesized from all regions.
(c) Construct the state graph of this elementary net system.
(d) Show that there is a smallest admissible set of regions R.
(e) Compare the net system synthesized from R with the net system synthesized from all minimal regions.



1.4.
(a) Show that states s_1 and s_2 cannot be separated by any region in any of the two initialized transition systems shown in Fig. 1.18 (page 31) .
(b) Show that event c cannot be separated from state s by any region in any of the two initialized transition systems shown in Fig. 1.19 (page 31).

1.5. Compute the regions of the transition systems shown in Fig. 1.13 and Fig. 1.26 (on the right hand side). Compute the net systems synthesized from minimal regions and their reachability graphs. Conclude that the given transition systems are not separated.

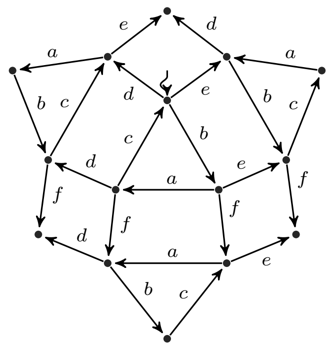
1.6. Let $\mathcal{U}(A) = (S', E, \delta', s'_0)$ be the limited unfolding of a finite initialized transition system $A = (S, E, \delta, s_0)$ (Def. 1.90 on page 50). Show that the simulation map $\varphi : L(A) \to \mathcal{U}(A)$ is the unique map $\varphi : L \to S'$ such that *(i)* $\varphi(u) = u$ if $u \in S'$, and *(ii)* $\varphi(u_1u_2u_3) = \varphi(u_1u_3)$ if $\delta(s_0, u_1u_2) = \delta(s_0, u_1)$.

1.7. The synchronized product of an i-indexed family of initialized transition systems $A_i = (S_i, E, \delta_i, s_{0,i})$, $i \in I$, is the initialized transition system $A = (S, E, \delta, s_0)$ where s_0 is the i-indexed vector with entries $s_0(i) = s_{0,i}$ and S and δ are defined simultaneously from the axiom $s_0 \in S$ by the inductive statement: $(\forall s \in S)\ \ (\delta(s,e) = s' \ \wedge\ s' \in S \quad \text{if} \quad (\forall i \in I)\ \ (\delta_i(s(i), e) = s'(i))$. Show that a synchronized product of separated initialized transition systems is separated.

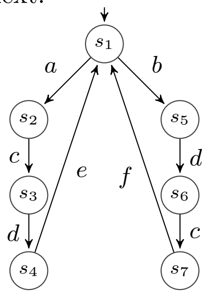
1.8. Write a program taking as argument a set X of subsets of a set R and computing all minimal subsets Y of R that intersect every element of X. Apply this program to the set R of non trivial regions of the four season transition system (Fig.1.21) in order to compute all minimal admissible sets of regions and all minimal strongly admissible sets of regions (yielding realizations of the transition system by elementary net systems and by contact-free elementary net systems, respectively).

1.9 (From [16]). Two states s and s' of an initialized transition system are said to be confluent if at least one state is reachable both from s and from s'. An initialized transition system is said to be confluent (resp. conditionnally confluent) if all pairs of states (resp. all pairs of language equivalent states) are confluent. Thus in particular, a finite initialized transition system in which the initial state may be reached from any other state is confluent. Prove the following statements:
(a) Two language equivalent states of a confluent (or conditionally confluent) initialized transition system cannot be separated by any region.
(b) If an initialized transition system A is confluent (or conditionally confluent) and it enjoys event-state separation, then $RG(SN(A))$ is a minimal automaton.
(c) If an initialized transition system A enjoys event-state separation, then any two states of A which cannot be separated by regions are language equivalent.

1.10.
(a) Compute the regions of the transition system shown on the right.
(b) Verify that all non trivial regions are incomparable for set inclusion.
(c) Extract a minimal admissible set of regions and construct the corresponding synthesized net system.
(d) Describe all the minimal admissible sets of regions of this transition system.



1.11. For the initialized transition system A shown next:
(a) Compute the set $Rmin(A)$ of minimal regions (as defined in Exer. 1.2).
(b) Whenever a unique minimal region separates strongly an event e from a state s, this region must belong to every strongly admissible set of minimal regions. Find five such minimal regions.



(c) Show that there are three distinct minimal strongly admissible sets of minimal regions.

2

Algorithms of Elementary Net Synthesis

In this chapter, we consider *finite* transition systems exclusively. The chapter contains three sections. The first section focusses on minimal regions and shows that they are *sufficiently complete* for all forms of the net synthesis problem described in Chap. 1, i.e., whenever a set of regions is admissible for some separation property, some set of minimal regions is also admissible. The interest is to reduce significantly the search space for admissible sets of regions, which has a direct impact on the efficiency of the net synthesis algorithms. The second section shows that unfortunately, the net synthesis problem is NP-complete, hence one cannot construct very efficient synthesis algorithms. The third section constructs a flexible algorithm, based on minimal regions, that can be tailored to all forms of the net synthesis problem by selecting the relevant separation axioms. Most developments presented in the chapter, in spite of the title, are in fact independent of elementary nets and valid in the larger framework of quasi-elementary nets. By nets and net systems, reconducting the convention adopted in Chap. 1, we always mean quasi-elementary nets and quasi-elementary net systems, thus *elementary* is never meant unless it is explicit.

2.1 Minimal Regions and State Machine Decompositions

Given an initialized transition system A, deciding whether A can be realized exactly by a net system and providing it with a net realization amounts to searching for an admissible set of regions R and synthesizing the net system $SN_R(A)$ induced from R. For any two distinct states, or state and event disabled at this state, one should search for a region of A enforcing the property of separation SSP, or ESSP, respectively. As far as theory is concerned, this search makes no difficulty: A is finite and $R(A)$ is also finite, hence it is straightforward to design an effective decision and realization procedure for the basic net synthesis problem. However, the set of regions of an initialized transition system with set of states S may be as large as 2^S. This would not be a problem if it was possible, for each pair of states or state and event to be separated, to compute separating regions with an efficient algorithm. Unfortunately, no such algorithm is known, and one is forced to *search* for separating regions instead of *computing* them. It is therefore highly relevant, in order to obtain reasonably efficient net synthesis procedures, to try reducing the search space to smaller subsets of regions with easy criteria of recognition. Minimal regions were put forward for this purpose by Bernardinello in [5], and they have been widely used since then, e.g., in the tool Petrify whose principles will be presented in the end of Sec. 2.3.

The rest of the section is organized as follows. First, we establish the crucial property of regions to be closed under relative complement. On this basis, we show that minimal regions are sufficiently complete for the basic net synthesis problem, i.e., an initialized transition system has an admissible set of regions if and only if it has an admissible set of minimal regions. So, it is not necessary to compute non-minimal regions for solving the basic net synthesis problem. Second, we show that net versions of initialized transition systems synthesized from (regions or) minimal regions may always be covered by state machine components, hence net synthesis reveals implicit concurrency. We give examples showing that unfortunately, in some cases where the basic net synthesis problem has feasible solutions, one cannot find least admissible sets of minimal regions solving this problem. Much freedom is therefore left for the design of net synthesis procedures, even if they use only minimal regions. Next, we show that for any initialized transition system, separated or non-separated, the net systems synthesized from all minimal regions and from all regions, respectively, are equivalent. So, it is not necessary to compute non-minimal regions for obtaining the best over-approximation of a transition system by the reachability graph of a net system. We finally show that minimal regions are sufficiently complete also for net synthesis from transition systems up to folding operations preserving languages, and for net synthesis from transition systems up to language equivalence. Hence, minimal regions are sufficiently complete for all forms of the net synthesis problem described in Chap. 1.

2.1.1 Minimal Regions are Sufficient for Synthesis

We will show in this section that minimal regions are sufficiently complete for the basic net synthesis problem. In other words, one can design correct decision and realization procedures for the basic net synthesis problem in which the search space for regions is reduced to the minimal regions, thus possibly increasing efficiency. The crux of the development presented in this part is the following proposition.

Proposition 2.1. *Let A be an initialized transition system. Let r, r' be two regions of A. If $r' \subseteq r$, then $r \setminus r'$ is a region of A.*

Proof. Three mutually exclusive cases can occur for an event e (see Fig. 2.1).

1. All occurrences of e enter r.
   Then either all occurrences of e enter r', and therefore they do not cross the border of $r \setminus r'$, or no occurence of e crosses the border of r', and therefore all occurrences of e enter $r \setminus r'$.
2. All occurrences of e exit from r.
   Then either all occurrences of e exit from r', and therefore they do not cross the border of $r \setminus r'$, or no occurence of e crosses the border of r', and therefore all occurrences of e exit from $r \setminus r'$.
3. No occurrence of e crosses the border of r.
   Then three mutually exclusive sub-cases can occur.
   a) All occurrences of e enter r'.
      Then all occurences of e exit from $r \setminus r'$.
   b) All occurrences of e exit from r'.
      Then all occurences of e enter $r \setminus r'$.
   c) No occurrence of e crosses the border of r'.
      Then no occurrence of e crosses the border of $r \setminus r'$.

□

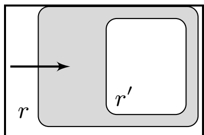
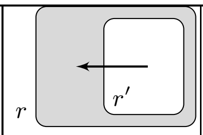
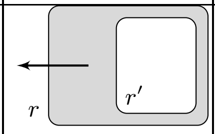
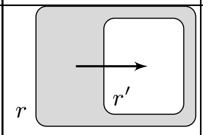
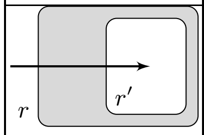
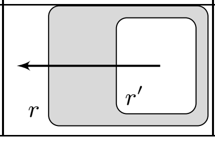
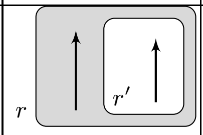
$e \in {}^{\circ}r$	$e \in r^{\circ}$	$e \in r^{\perp}$	
			$e \in {}^{\circ}(r \setminus r')$
			$e \in (r \setminus r')^{\circ}$
			$e \in (r \setminus r')^{\perp}$

Table 2.1. a graphical proof of Prop. 2.1

As we consider finite transition systems exclusively, every initialized transition system A has a non-empty set of minimal non-empty regions. For convenience, let us introduce a notation for this set.

Definition 2.2. *Given an initialized transition system A, let $Rmin(A)$ denote the subset of regions of A which are minimal w.r.t. set inclusion in $R(A) \setminus \emptyset$.* ♢

In the sequel, *minimal region* means always *minimal non-empty region*. Three lemmas based on Prop. 2.1 are needed to show that, if an initialized transition system has an admissible set of regions, then it has an admissible set of minimal regions.

Lemma 2.3. *Let A be an initialized transition system. Let r be a region of A. Let s, s' be two states of A, such that $s \in r$ and $s' \notin r$. Let r' be a region of A strictly included in r. Then r' or $r \setminus r'$ separates state s from state s'.*

Proof. As r' and $r \setminus r'$ form a partition of r it follows from $s' \notin r$ that $s' \notin r'$ and $s' \notin r \setminus r'$ and from $s \in r$ that $s \in r'$ or $s \in r \setminus r'$. □

Lemma 2.4. *Let A be an initialized transition system with set of states S. Let s, r, and e be a state, a region, and an event of A, respectively, such that r separates event e from state s. Then either e exits from r and $s \notin r$, or e exits from the complementary region $S \setminus r$ and $s \notin (S \setminus r)$.*

Proof. By Def. 1.42, if r separates event e from state s, then either e exits from r and $s \notin r$, or e enters r and $s \in r$. In the latter case, necessarily, the event e exits from the complementary region $S \setminus r$ and $s \notin (S \setminus r)$. □

Lemma 2.5. *Let A be an initialized transition system. Let s, r, and e be a state, a region, and an event of A, respectively, such that e exits from r and $s \notin r$. Let r' be a region of A strictly included in r. Then either r' or $r \setminus r'$ separates event e from state s.*

Proof. In view of Table 2.1 every event that exits from r exits either from r' or from $r \setminus r'$. A state that does not belong to r belongs neither to r' nor to $r \setminus r'$, which are subsets of r, hence either r or $r \setminus r'$ separates event e from state s. □

We can now show that any separated transition system (see Def. 1.46) may be realized by the net system synthesized from its minimal regions.

Proposition 2.6. *Let A be an initialized transition system. If A is separated, then $A \cong RG(SN_R(A))$ for $R = Rmin(A)$.*

Proof. As A is a separated transition system, the set $R(A)$ of all regions of A is admissible. As $R(A)$ is admissible, by Lemma 2.3, every pair of states is separated by some region in $Rmin(A)$. As $R(A)$ is admissible, by Lemma 2.4, every event e disabled at some state s is separated from this state by some pre-region $r \in {}^\circ e$. By Lemma 2.5, the event e is then disabled at s by some region in $Rmin(A)$. Therefore, $Rmin(A)$ is an admissible set of regions of A, and by Th. 1.43, $A \cong RG(SN_R(A))$. □

2.1.2 Regions and State-Machine Decompositions

We will show in this section that the net systems synthesized from all regions of an initialized transition system, or from all minimal regions of an initialized transition system, may be covered by sequential components. The import of this result is to give evidence of the fact that net synthesis *extracts concurrency* from (sequential) transition systems, in which it is implicit. The state machine decomposition of net systems synthesized from minimal regions relies essentially on Prop. 2.1 which was established in the last section.

Before we introduce definitions and terminology about state machine components and state machine decompositions, we need to define subnet systems.

Definition 2.7. *Let $N = (P, T, F, M_0)$ be a net system. A* subnet system *of N is a net system $N' = (P', T', F', M_0')$ such that the following conditions hold:*

1. *$P' \subseteq P$ and $T' \subseteq T$,*
2. *$\forall p \in P' \; \forall t \in T \;\; ((p,t) \in F \vee (t,p) \in F) \Rightarrow t \in T'$,*
3. *F' is the induced restriction of F on $(P' \times T') \cup (T' \times P')$,*
4. *M_0' is the induced restriction of M_0 on P'.*

A subnet system N' of N is connected *if the graph $(P' \cup T', F')$ is connected.*

This definition is more demanding than other definitions of subnets sometimes found in the literature, because it requires that transitions connected with a place come along with that place into the subnet.

Definition 2.8. *Let $N = (P, T, F, M_0)$ be a net system. A* state machine component *of N is a connected subnet system N' of N, let $N' = (P', T', F', M_0')$, such that $|M_0'| = 1$ (i.e., there is exactly one place marked in M_0') and for every transition $t' \in T'$, $|{}^\bullet t'| = 1$, and $|t'^\bullet| = 1$.*

Example 2.9. The elementary net system for mutual exclusion shown in Fig. 1.5 has exactly three state machine components, shown in Fig. 2.1.

□

A state-machine $N' = (P', T', F', M_0')$ may be seen alternatively as an initialized transition system (P', T', δ', p_0') with the set of states P', the initial state $p_o' \in M_0'$, and the transition map $\delta'(p', t') = p"$ if ${}^\bullet t' = \{p'\}$ and $t'^\bullet = \{p"\}$. In fact, the reachability graph of N' is isomorphic to this initialized transition system. Every state-machine $N' = (P', T', F', M_0')$ is therefore a contact-free net system.

Definition 2.10. *A* state machine decomposition *of N is a family of state machine components $N_i = (P_i, T_i, F_i, M_{0,i})$ of N that* covers *N, i.e., such that $P = \cup_i P_i$ and $T = \cup_i T_i$ where P and T are the subsets of non-isolated places and transitions of N, respectively.* ♢

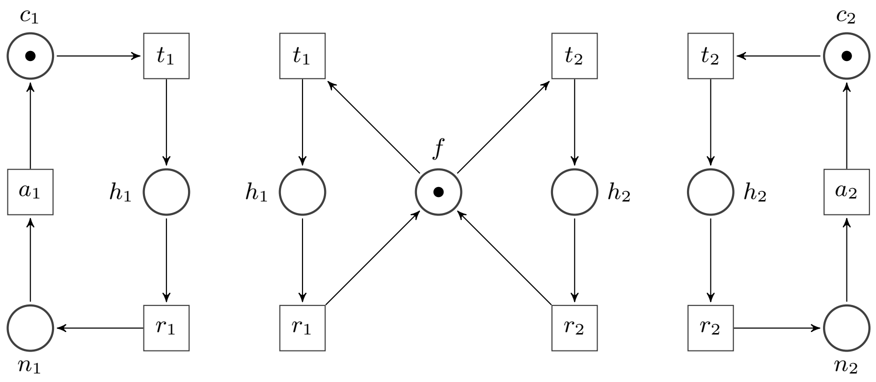


Fig. 2.1. the three state-machine components of the net system in Fig. 1.5

Example 2.11. The state-machines shown in Fig.2.1 cover the net system of Fig. 1.5 and provide a state machine decomposition of this system. The three state-machines represent the two process components and the resource, taken in isolation. □

A net system N in which every place p has a complementary place p' may always be decomposed into state machine components N_i, each of which is determined by setting $P_i = \{p, p'\}$ for some place p of N. As adding complementary places does not affect the behaviour of net systems, the reachable state graph $RG(N)$ of a net system N is always isomorphic to the reachable state graph $RG(N')$ of some contact-free and state-machine decomposable net system N' [32].

Let us consider now the particular case of the net systems $SN(A)$ ($= SN_{R(A)}(A)$) or $SN_{Rmin(A)}(A)$ synthesized from all regions or from all minimal regions of an initialized transition system A. We want to show that both net systems may be covered by state-machine components (plus possibly isolated places and transitions). The proof of this result depends mainly upon Prop. 2.1, which shows that any non-minimal region of an initialized transition system A may be *partitioned* into smaller regions. Indeed, if r is a non-minimal region of A, then there must exist some minimal region $r' \subseteq r$, and by Prop. 2.1, the relative complement $r \setminus r'$ is also a region of A. However, in addition to Prop. 2.1, we need the weakly converse property as follows.

Proposition 2.12. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system. Let r_1, r_2 be two disjoint regions of A. Then $r_1 \cup r_2$ is a region of A.*

Proof. Let $e \in E$. If all occurences of e enter r_1, then either they all exit from r_2 and therefore they do not cross the border of $r_1 \cup r_2$, or none of them crosses the border of r_2 and therefore they all enter $r_1 \cup r_2$. If all occurences of e exit from r_1, then symmetrically, either they do not cross the border of $r_1 \cup r_2$, or they all exit from $r_1 \cup r_2$. Similar considerations apply to r_2. The remaining case is when no occurence of e crosses the borders of r_1 and r_2 and then no occurrence of e crosses the border of $r_1 \cup r_2$. □

Take any initialized but not necessarily separated transition system $A = (S, E, \delta, s_0)$. By Prop. 2.1, the set of states S may be partitioned into regions (or into minimal regions) of A and every non-empty region (or minimal region) of A appears in at least one such partition. The next proposition states that every non-trivial partition of S into regions of A determines a state-machine component of the saturated net system synthesized from the regions of A. In particular, every non-trivial partition of S into minimal regions of A determines a state-machine component of the net system synthesized from the minimal regions of A.

Proposition 2.13. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system. Let $R = \{r_1, \ldots, r_n\}$ be a non-trivial partition of S into regions of A. Let $T = \cup_i {}^\circ r_i$. Then for every $t \in T$, $|{}^\circ t \cap R| = 1$, and $|t^\circ \cap R| = 1$. Moreover, the net system (R, T, F, M_0) defined by $r_i \in M_0$ iff $s_0 \in r_i$, $(r_i, t) \in F$ iff $t \in r_i{}^\circ$, and $(t, r_i) \in F$ iff $t \in {}^\circ r_i$, is connected.*

Proof. Let $t \in T$. By construction of T, all occurences of the event t in A enter jointly some region $r_i \in R$, necessarily unique. As R is a partition of S, all occurences of the event t in A exit from some other region $r_j \in R$, necessarily unique. As t enters r_i and t exits from r_j, $r_i \neq r_j$ be definition of regions. We now prove that $N = (R, T, F, M_0)$ is a connected net. Consider any split of R into non-empty and complementary subsets $\{r_1, \ldots, r_k\}$ and $\{r_{k+1}, \ldots, r_n\}$. By Prop.2.12, $r = \cup_{i=1}^{k} r_i$ and $r' = \cup_{i=k+1}^{n} r_i$ are two (complementary) non-trivial regions of A. As r is non-trivial, $e \in r^\circ$ for some $e \in E$, and since r and r' are complementary regions, $e \in {}^\circ r'$. Therefore, all occurrences of e in A exit jointly from some region r_i with index $i \leq k$ and they enter jointly some other region r_j with $j \geq k+1$. As a result, $(r_i, e) \in F$ and $(e, r_j) \in F$. As the split of R was arbitrary, N is connected. □

Proposition 2.13 establishes the following two claims:

1. For every initialized transition system A with set of states S, the saturated net system $SN(A)$ synthesized from the regions of A may be covered by state-machine components, induced from non-trivial partitions of S into regions of A.
2. For every initialized transition system A with set of states S, the net system $SN_{Rmin(A)}(A)$ synthesized from the minimal regions of A may be covered by state-machine components, induced from non-trivial partitions of S into minimal regions of A.

As every state-machine component of a net system is contact free, the net systems $SN(A)$ and $SN_{Rmin(A)}(A)$ are contact-free.

2.1.3 Minimal Admissible Sets of Regions

In Section 1.4, the four season example has shown that there may exist minimal *admissible* sets of regions which are incomparable w.r.t. set inclusion. Moreover, these minimal admissible sets of regions may have different cardinalities. Similar observations are valid if one replaces admissible sets of regions with *strongly admissible* sets of regions. In this section, we show that the situation is not improved if one replaces regions with *minimal regions*, hence searching for minimal admissible sets of minimal regions cannot be the target of the design of net synthesis algorithms.

In the four season example, there was only one admissible set of minimal regions, and the following questions could not be settled:

- May there exist several minimal admissible sets of minimal regions?
- If so, do all minimal admissible sets of minimal regions have the same cardinality?

We address these two questions using an example borrowed from [4].

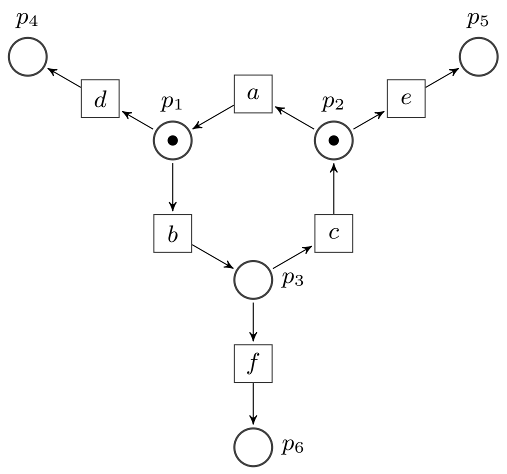


Fig. 2.2. an elementary net system

Example 2.14. Consider the elementary net system N shown in Fig. 2.2. The reachability graph $RG(N)$ of this net system is shown in Fig. 2.3, where the markings are written as ordered sequences of 0 and 1 following the natural order on the places (the initial marking is 110000). This elementary transition system has 12 non-trivial regions, namely the regions r_i and r_i' defined for $i = 1, \ldots, 6$ as follows: let r_i be the set of reachable markings of N with the i^{th} digit equal to 1, i.e. $r_i = \{M \mid p_i \in M\}$, and let r_i' be the set of reachable

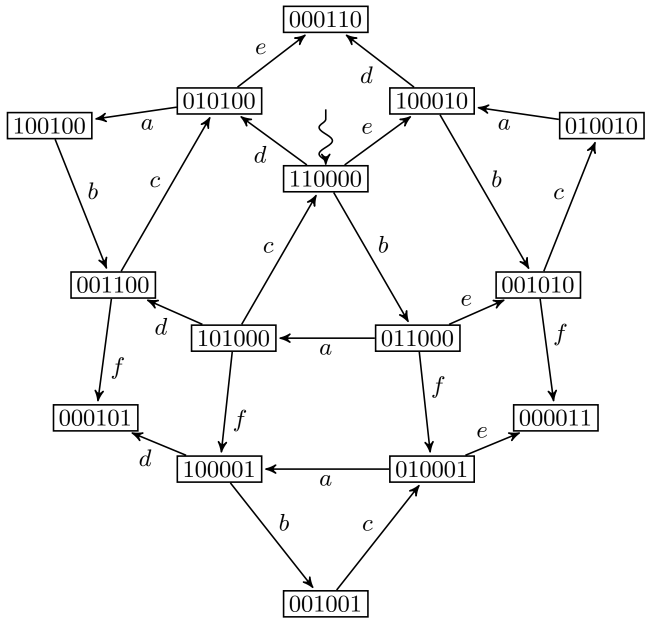


Fig. 2.3. the reachability graph of the elementary net system in Fig. 2.2

markings of N with the i^{th} digit equal to 0: $r'_i = \{M \mid p_i \notin M\}$. As r_i and r'_i are complementary, there exist exactly six partitions of the set of states of $RG(N)$ into regions. One can check that all the considered regions are minimal, hence every proper partition of the set of states of $RG(N)$ into regions is a partition into minimal regions. By the way, the minimal regions r_1, r_2, r_3 have an empty intersection although they intersect pairwise. A minimal admissible set of minimal regions is $R = \{r_1, r_2, r_3, r_4, r_5, r_6\}$, as the reader may check. R is not strongly admissible, since the net system $SN_R(RG(N))$ synthesized from R is isomorphic to the net system N and N has contacts (e.g., a cannot be fired in the initial marking because p_1, which is a post-place of a, is marked). Now seeing that r_i and r'_i are complementary regions, any set R' obtained by replacing some or all of the regions r_i by their complement r'_i is also a minimal admissible set of minimal regions. Moreover all minimal admissible sets of minimal regions are obtained in this way. Indeed, a minimal set of minimal regions cannot contain complementary regions since two complementary regions are equivalent w.r.t. the separation properties SSP and ESSP. THis example thus provided an initialized transition system with many minimal admissible sets of minimal regions. □

In the above example, all minimal admissible sets of minimal regions have the same cardinality. This may be changed by adding to the transition system shown in Fig. 2.3 a new state s_0, taken as the initial state, and three transitions $s_0 \xrightarrow{i} 110000$, $s_0 \xrightarrow{j} 101000$, and $s_0 \xrightarrow{k} 011000$. Let A be the resulting initialized transition system. The minimal regions of A are r_i and r'_i defined as above ($i = 1, \ldots, 6$) plus the minimal region $r_7 = \{s_0\}$. Remark that r_i and r'_i are no longer complementary in A. The set of states of A has now six partitions $\{r_i, r'_i, r_7\}$ into minimal regions. $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$ is a minimal admissible set of minimal regions, but $R' = \{r'_1, r'_2, r'_3, r'_4, r'_5, r'_6\}$ is also a minimal admissible set of minimal regions. Therefore, *minimal admissible sets of minimal regions may have different cardinalities.*

The questions of uniqueness or equal cardinality of minimal *strongly* admissible sets of minimal regions may be asked similarly. The example presented above does not help answering these questions, because A has a unique minimal strongly admissible set of minimal regions, namely $\{r_1, r_2, r_3, r'_1, r'_2, r'_3, r'_4, r'_5, r'_6, r_7\}$. However, the next example, borrowed from [17], shows that *minimal strongly admissible sets of minimal regions are generally not unique.* So, the only question left open is whether all minimal strongly admissible sets of minimal regions have equal cardinality.

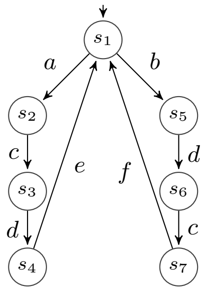


Fig. 2.4. an elementary transition system

Example 2.15. Consider the elementary transition system shown in Fig. 2.4. All minimal pre-regions of the events that occur in this elementary transition system are listed in the next table.

pre-region of	events
$r_0 = \{s_2, s_5, s_6\}$	c
$r_1 = \{s_2, s_4, s_6\}$	c, e
$r_2 = \{s_2, s_3, s_5\}$	d
$r_3 = \{s_2, s_3, s_4\}$	e
$r_4 = \{s_3, s_5, s_7\}$	d, f
$r_5 = \{s_5, s_6, s_7\}$	f
$r_6 = \{s_3, s_4, s_7\}$	e, f
$r_7 = \{s_4, s_6, s_7\}$	e, f
$r_8 = \{s_1\}$	a, b

Note that, whenever a unique minimal region separates strongly an event e from a state s, this region must belong to every strongly admissible set of minimal regions. Using this argument, one can infer that every strongly admissible set of minimal regions of the transition system shown in Fig. 2.4 contains the set $\{r_0, r_1, r_2, r_4, r_8\}$. The reader may check that by adding to this set any of the sets $\{r_3, r_7\}$ or $\{r_5, r_6\}$ or $\{r_6, r_7\}$, one obtains a minimal strongly admissible set of minimal regions. So, such sets are generally not unique. □

2.1.4 Minimal Regions are Sufficient for Approximate Synthesis

In Sec. 2.1.1, we have shown that minimal regions are sufficiently complete for the basic net synthesis problem. We will show that minimal regions are sufficient also for computing optimal solutions to the approximate net realization problem. Recall from Section 1.5 that for any initialized transition system A, the net system $SN(A)$ synthesized from all regions of A yields an optimal over-approximation of A by the reachability graph of a net system ($A \leq RG(N)$ *iff* $N \leq SN(A)$). We aim at proving that the net system $SN(A)$ synthesized from all regions of A and the net system $SN_{Rmin(A)}(A)$ synthesized from all minimal regions of A have in fact isomorphic reachability graphs. One can therefore design optimal procedures for the approximate net realization problem in which the search space for regions is reduced to minimal regions, thus resulting in *exponentially smaller* net systems.

In the sequel, $A = (S, E, \delta, s_0)$ is an initialized transition system, and $\eta_A : A \to RG(SN(A))$ is the simulation map that sends each state of A to the marking of $SN(A)$ comprised of the regions of A which this state belongs to (see Def. 1.76). Let us recall that for any region $r \in RG(SN((A))$, $\eta_A^{-1}(r)$ is a region of A with the same signature as r (Lemma 1.56). A main step towards proving that $SN(A)$ and $SN_{Rmin(A)}(A)$ have isomorphic reachability graphs (Prop. 2.18) is to show that $\eta_A^{-1}(r)$ restricts to a bijection between $Rmin(RG(SN(A)))$ and $Rmin(A)$. For the ease of the presentation, in order to prove this intermediate result, we shall temporarily accept without proof the following statement, which will be established after the main proposition.

Lemma 2.16. *The map* $\eta_A^{-1} : R(RG(SN(A))) \to R(A)$ *is a bijection between regions, and it preserves and reflects disjointness of regions.*

As the region $\eta_A^{-1}(r)$ of A has the same signature as the region r of $RG(SN(A))$, Lemma 2.16 tells us among other that the map η_A^{-1} (preserves and) reflects signatures of regions.

Lemma 2.17. *The induced restriction of* η_A^{-1} *on* $Rmin(RG(SN(A)))$ *is a bijection between* $Rmin(RG(SN(A)))$ *and* $Rmin(A)$.

Proof. The proof proceeds by contradiction.

1. Suppose $\eta_A^{-1}(r)$ is a minimal region of A but r is a non-minimal region of $RG(SN(A))$. By Prop. 2.1, $r = r_1 \cup r_2$ is the union ot two non-empty and disjoint regions of $RG(SN(A))$. By definition of inverse maps, $\eta_A^{-1}(r) = \eta_A^{-1}(r_1) \cup \eta_A^{-1}(r_2)$. By Lemma 2.16, $\eta_A^{-1}(r_1)$ and $\eta_A^{-1}(r_2)$ are two non-empty and disjoint regions of A, both included in $\eta_A^{-1}(r)$, hence $\eta_A^{-1}(r)$ was not minimal.
2. Suppose r is a minimal region of $RG(SN(A))$ but $\eta_A^{-1}(r)$ is a non-minimal region of A. By Prop. 2.1, $\eta_A^{-1}(r) = r_1' \cup r_2'$ is the union of two non-empty and disjoint regions of A. As η_A^{-1} is a bijection between the regions of $RG(SN(A))$ and the regions of A, $r_1' = \eta_A^{-1}(r_1)$ and $r_2' = \eta_A^{-1}(r_2)$ for some non-empty regions r_1, r_2 of $RG(SN(A))$. As η_A^{-1} reflects disjointness, r_1 and r_2 are disjoint. By Prop. 2.12, $r_1 \cup r_2$ is a region of $RG(SN(A))$. Now $\eta_A^{-1}(r_1 \cup r_2) = \eta_A^{-1}(r_1) \cup \eta_A^{-1}(r_2) = r_1' \cup r_2' = \eta_A^{-1}(r)$, and because η_A^{-1} is a bijection, $r = r_1 \cup r_2$, contradicting the assumption that r is a minimal region. □

We can now state and prove the main result of the section.

Proposition 2.18. *For any initialized transition system A, the net systems $SN(A)$ and $SN_{Rmin(A)}(A)$ have isomorphic reachability graphs.*

Proof. Let $B = RG(SN(A))$. By Th. 1.77, B is a separated transition system. By Prop. 2.6, $B \cong RG(SN_{Rmin(B)}(B))$, i.e., B is realized exactly by the net synthesized from its minimal regions. Therefore, $RG(SN(A)) \cong RG(SN_{Rmin(B)}(B))$. By Lemma 2.17, η_A^{-1} maps bijectively $Rmin(B)$ to $Rmin(A)$ such that two corresponding regions have the same signature, hence the net systems $SN_{Rmin(A)}(A)$ and $SN_{Rmin(B)}(B)$ are isomorphic. The net systems $SN(A)$ and $SN_{Rmin(A)}(A)$ have therefore isomorphic reachability graphs. □

It remains to establish Lemma. 2.16.

Proof (of Lemma 2.16).
We show first that $\eta_A^{-1} : R(RG(SN(A))) \to R(A)$ is a bijection. As η_A^{-1} preserves signatures of regions, it suffices, in view of Lemma 1.32, to show that every signature of a region of A coincides with the signature of some region of $RG(SN(A))$. Indeed, for any region of A, the signature of this region is the same as the signature of the place of $SN(A)$ induced from this region, and for every place of $SN(A)$, the signature of this place is the same as the signature of the region of $RG(SN(A))$ defined as its extension. Therefore, η_A^{-1} is a bijection. $\eta_A^{-1} : R(RG(SN(A))) \to R(A)$ obviously preserves disjointness of regions since it is defined as an inverse map. In order to complete the proof of the lemma, we show that for any two regions r_1, r_2 of $RG(SN(A))$, if $\eta_A^{-1}(r_1) \cap \eta_A^{-1}(r_2) = \emptyset$, then r_1 and r_2 are disjoint. As $\eta_A^{-1}(r_1)$ and $\eta_A^{-1}(r_2)$ are disjoint, by Propositions 2.12 and 2.1, the set $S \setminus (\eta_A^{-1}(r_1) \cup \eta_A^{-1}(r_2))$ is a region of A, let $\eta_A^{-1}(r_3)$ where $r_3 \in R(A)$. By Prop. 2.13, the partition

$\{\eta_A^{-1}(r_1), \eta_A^{-1}(r_2), \eta_A^{-1}(r_3)\}$ of S determines a state-machine component of the net system $SN(A)$, i.e., exactly one of the places p_1, p_2, p_3 of $SN(A)$ induced from these three regions of A is marked in any reachable marking of $SN(A)$. Now for each i, the place p_i has the same signature as the region $\eta_A^{-1}(r_i)$ from which it has been induced, and the place p_i has the same signature as its extension r'_i which is a region of $RG(SN(A))$. As $\eta_A^{-1} : R(RG(SN(A))) \to R(A)$ is a bijection preserving signatures of regions, necessarily, $r'_i = r_i$. As p_1, p_2, p_3 form a sequential component of $SN(A)$, the regions r_1, r_2, r_3 are disjoint, hence in particular r_1 and r_2 are disjoint. □

2.1.5 Minimal Regions are Sufficiently Complete for Synthesis up to Folding Operations Preserving Languages and for Synthesis up to Language Equivalence

According to Prop. 1.81, an initialized transition system A may be realized by an elementary net system up to folding operations preserving languages if and only if the axioms of event-state separation, event effectiveness, and event simpleness are satisfied in A. In this case, $A \rhd RG(SN_R(A))$ for any set of regions $R \subseteq R(A)$ enforcing these properties. In this section, we show that the set of minimal regions $Rmin(A)$ is *sufficiently complete* w.r.t. all three axioms, which means that whenever the axioms are satisfied for the set $R(A)$, they are satisfied also for the set $Rmin(A)$. This result entails sufficient completeness of minimal regions for the synthesis of net systems up to language equivalence. Indeed, by Prop. 1.97, the synthesis of a net system from an initialized transition system A up to language equivalence reduces to the synthesis of a net system from the limited unfolding $\mathcal{U}(A)$ of A up to folding operations preserving languages. The minimal regions of a (prefix-closed) regular language are therefore sufficiently complete for the realization of this language by an elementary net system.

The proposition stated and established below completes our efforts to show that minimal regions are sufficiently complete for all forms of the net synthesis problem, explaining why Section 2.3 will be focussed on the search for minimal regions exclusively.

Proposition 2.19. *Minimal regions are sufficiently complete w.r.t. the axioms of event-state separation, event effectiveness, and event simpleness.*

Proof. As regards event-state separation, Lemmas 2.4 and 2.5 have already shown in Sec. 2.1.1 that minimal regions are sufficiently complete. We address now the axioms of event effectiveness and event simpleness. Let e be an event with an input region r. For every region $r' \subset r$, either r' or $r \setminus r'$ is an input region of e. It follows by induction on finite sets that e has some minimal input region included in r. As regards event simpleness, we show that any non minimal region r separating two events e and e' contains a strict subregion

$r' \subset r$ which separates e and e' as well. Suppose, as a first case, that events e and e' are distinguished by some input region, e.g., $r \in {}^{\circ}e$ and $r \notin {}^{\circ}e'$. Since r is not minimal, there exists a region $r' \subset r$ and either r' or $r \setminus r'$ is an input region of e. Suppose for instance that $r' \in {}^{\circ}e$ (and thus $r \setminus r' \in e^{\perp}$). If $r' \notin {}^{\circ}e'$, then r' separates e from e'. If $r' \in {}^{\circ}e'$, then for every event e', $s \xrightarrow{e'} s' \Rightarrow (s \in r' \ \wedge \ s' \notin r')$, and $(r' \subset r \ \wedge \ r \notin {}^{\circ}e') \Rightarrow (s \in r \ \wedge \ s' \in r)$, hence altogether $(s \notin (r \setminus r') \ \wedge \ s' \in (r \setminus r'))$. Therefore in this case, $(r \setminus r') \in e^{\perp}$ and $(r \setminus r') \in e'^{\circ}$, showing that $r \setminus r'$ separates e from e'. The case where events e and e' are distinguished by output regions can be treated similarly: one can apply the above reasoning on the dual transition system obtained by reversing all transitions. In fact, if let the dual (S, E, Δ^{opp}) of the transition system (S, E, Δ) be defined by $(s, e, s') \in \Delta^{opp} \Leftrightarrow (s', e, s) \in \Delta$, then (S, E, Δ) and (S, E, Δ^{opp}) have the same regions $r = r^{opp} \subseteq S$, where $r \in {}^{\circ}e$ iff $r^{opp} \in e^{\circ}$ and $r \in e^{\circ}$ iff $r^{opp} \in {}^{\circ}e$. Thus, in both cases, we can find a region r' strictly included in r separating e and e'. It follows by induction on finite sets that e and e' may be separated by a minimal region. □

2.2 NP-Completeness of Synthesis

Before we deal with the design of net synthesis algorithms, it is worth studying the complexity of the net synthesis problem, since it determines more or less which types of techniques can be applied in such algorithms. The analysis done in this section shows that elementary net synthesis algorithms rely on purely combinatorial methods.

In the section, the regions $r \subseteq S$ of an initialized transition system $A = (S, E, \delta, s_0)$, and more generally all subsets $r \subseteq S$, are identified with their characteristic functions. By Prop. 1.40, an elementary transition system $A = (S, E, \delta, s_0)$ has always some admissible set of regions R with size $|R|$ less than or equal to $|S| \times ((|S| - 1)/2 + |E|)$, where $|E| \leq |S| \times (|S| - 1)$ since elementary transition systems are loop-free and simple. Every region $r \in R$, as a subset of S, has size $|r|$ less than or equal to $|S|$. If the elementary net synthesis problem has a solution for an initialized transition system A, one can therefore construct such a solution from an admissible set of regions R with size polynomial in the size of A.

Consider any set R of subsets $r \subseteq S$. Verifying that a subset $r \subseteq S$ is a region of A takes time linear in the number of transitions of A (hence quadratic in $|S|$) since it suffices to check that all transitions $s \xrightarrow{e} s'$ labelled with the same event e determine the same difference $r(s') - r(s)$ in $\{-1, 0, 1\}$. Verifying that R is an admissible set of regions takes time polynomial in $|S|$ and $|R|$, since it suffices to check for $|S| \times ((|S| - 1)/2)$ pairs of states s, s' that $r(s) \neq r(s')$ for some $r \in R$, and to check for at most $|S|^2 \times (|S| - 1)$ triples of states $s, s', s"$ (such that $s' \xrightarrow{e} s"$ for some event e disabled in s) that $r(s) < r(s') - r(s")$ for some $r \in R$.

Altogether, *i)* the elementary net synthesis problem has a solution for a loop-free and simple initialized transition system $A = (S, E, \delta, s_0)$ if and only if one can produce non-deterministically a polynomial size description of a set of subsets of S that happens to be an admissible subset of regions of A, and *ii)* for arbitrary polynomial size descriptions of sets of subsets of S, checking whether they define an admissible subset of regions of A can be done using time polynomial in $|S|$ and the size of these descriptions. The elementary net synthesis problem for loop-free and simple initialized transition systems falls therefore in NP by definition of this complexity class (NP means Non-Deterministic Polynomial Time).

In the rest of the section, we provide some intuitions under the stronger claim, established in [2], that the elementary net synthesis problem (for loop-free and simple initialized transition systems) is NP-complete. This property means that for any decision problem in the class NP, let $(Q(x), X)$ where Q is a predicate in one variable x ranging over a set of strings X, there exists an algorithm α, that translates strings σ to loop-free and simple initialized transition systems $\alpha(\sigma)$, using time polynomial in the length of the input strings, such that $Q(\sigma)$ is true if and only if $\alpha(\sigma)$ is an elementary transition system. The algorithm α is then called a polynomial time reduction of the problem $(Q(x), X)$ to the elementary net synthesis problem.

In section 2.2.1, it is shown that for each instance of the state separation or event-state separation problem taken in isolation, deciding whether this instance can be solved by a separating region is an NP-complete problem. In section 2.2.2, we give a sketch of the proof that it is also an NP-complete problem to decide whether all instances of the state separation and event-state separation problems can be solved together by some admissible set of regions.

2.2.1 The Separation Problems are *NP*-Complete

Let us call *state separation problem* the question whether, given an initialized transition system $A = (S, E, \delta, s_0)$ and two states $s, s' \in S$, these two states are separated by some region of A. Similarly, let us call *event state separation problem* the question whether, given an initialized transition system $A = (S, E, \delta, s_0)$, an event $e \in E$ and a state $s \in S$, some region of A separates e from s. Clearly, both separation problems are in the class NP. Hiraishi proved in [25] that they are in fact NP-complete. For this purpose, he showed that a well-known NP-complete problem, namely the satisfaction problem 3-SAT, reduces in polynomial time to the state separation problem, which reduces in turn in polynomial time to the event state separation problem. The latter point is clear from the following remark: two states $s, s' \in S$ are separated by some region of A if and only if, in the initialized transition system $A' = (S, E', \delta', s_0)$ defined by $E' = E \cup \{e'\}$, $e' \notin E$, $\delta'(s, e) = \delta(s, e)$ for $e \in E$, and $\delta(s, e') = s'$, the event e' is separated from the state s' by some region (of A'). We present below the reduction of 3-SAT to the state separation problem. As Hiraishi noted, this reduction has no direct implication on the complexity of the elementary net synthesis problem.

Recall that 3-SAT is the problem whether, given a finite set V of boolean variables ($v \in V$) and a finite system C of disjunctive clauses over V, with exactly three literals (v or $\neg v$) per clause, there exists a truth assignment for V satisfying all clauses in C (e.g., $\neg v_1 \vee \neg v_2 \vee \neg v_3$ is satisfied if the value $false$ is assigned to at least one variable in the set $\{v_1, v_2, v_3\}$). This problem is NP-complete, see e.g. [24].

From V and C, Hiraishi constructs a transition system A by gluing on two shared states s_1 and s_2 two collections of transition systems A_v ($v \in V$) and A_c ($c \in C$) with sets of states pairwise disjoint except for states s_1 and s_2. For each variable $v \in V$, let A_v be the transition system shown in Fig. 2.5(a). For each clause $c = c_1 \vee c_2 \vee c_3$ in C, where each c_i is either a variable or the complement of a variable in V, let A_c be the transition system shown in Fig. 2.5(b), where c_y, c_z and nc_y, nc_z are fresh events local to A_c. A has size polynomial in the size of C, and its initial state is s_2.

$$\begin{tikzcd}
s_2 \arrow[r, "v"] & s_v \arrow[r, "\neg v"] & s_1
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
s_2 \arrow[r, "c_1"] & sc_1 \arrow[r, "c_y"] & sc_2 \arrow[r, "c_2"] & sc_3 \arrow[r, "c_z"] & sc_4 \arrow[r, "c_3"] & s_1 \arrow[dll, "c_y"'] \arrow[ddll, "c_z"] \\
 & & & sc_y \arrow[ulll, "nc_y"'] & & \\
 & & & sc_z \arrow[uulll, "nc_z"] & &
\end{tikzcd}$$

(b)

Fig. 2.5. the transition systems representing a variable (a) and a clause (b)

The NP-completeness of the state separation problem is established by the following proposition, that shows a reduction of 3-SAT to the state separation problem.

Proposition 2.20. *The problem 3-SAT has a solution for V and C if and only if the state separation problem has a solution for s_1, s_2 and A.*

Proof. ($\Leftarrow$) Suppose that the state separation problem has a solution for s_1, s_2 and $A = (S, E, \delta, s_2)$. As $R(A)$ is closed under complementation of regions, there must exist two maps $r : S \to \{0, 1\}$ and $r : E \to \{-1, 0, 1\}$ such that $r(s_2) = 0$, $r(s_1) = 1$, and $r(s') = r(s) + r(e)$ whenever $\delta(s, e) = s'$. By restricting r on A_c for a fixed $c \in C$, one can see the following. As $r(s_1) = 1$, $r(c_y)$ and $r(c_z)$ must belong to the set $\{-1, 0\}$. Therefore, $r(c_1) + r(c_2) + r(c_3) \geq r(c_1) + r(c_y) + r(c_2) + r(c_z) + r(c_3) = r(s_1) - r(s_2) = 1$, i.e., $r(c_1) + r(c_2) + r(c_3) \geq 1$. Therefore, $r(c_i) = 1$ for at least one literal c_1 or c_2

or c_3 in every clause $c \in C$. By restricting r on A_v for a fixed variable $v \in V$, one can see the following. As $r(s_1) - r(s_2) = 1$, one has either $r(v) = 1$ and $r(\neg v) = 0$ or $r(v) = 0$ and $r(\neg v) = 1$. Let $f : V \to \{0,1\}$ be the boolean valuation defined by $f(v) = r(v)$ for all v. The truth assignment represented by f ($f(v) = 1$ iff v is *true*) is then a solution of 3-*SAT* for V and C.

($\Rightarrow$) Suppose now that 3-*SAT* has a solution for V and C, and let $f : V \to \{0,1\}$ be the boolean representation of this solution. Let $r(s_2) = 0$ and $r(s_1) = 1$. For every $v \in V$, let $r(v) = f(v)$ and $r(\neg v) = 1 - f(v)$. Each c_i is either a variable v or the negation $\neg v$ of a variable, hence $r(c_i) \in \{0,1\}$. For every $c \in C$, let $r(sc_1) = r(s_2)+r(c_1)$ and symmetrically $r(sc_4) = r(s_1)-r(c_3)$, hence $r(sc_1)$ and $r(sc_4)$ both belong to the set $\{0,1\}$. We want to show that one can complete the definition of r on the local states and events of each transition system A_c so that $r(s') = r(s) + r(e)$ whenever $\delta(s,e) = s'$ in A_c, hence yielding a region r of A that separates s_1 from s_2. Remark that for any such region, $r(c_y)$ and $r(c_z)$ belong necessarily to the set $\{-1,0\}$ since $r(s_1) = 1$.

The rows of the following table enumerate the eight possibilities for the values (in $\{0,1\}$) of $r(sc_1)$, $r(c_2)$ and $r(sc_4)$ (grey columns of the table). In each row, the other entries are filled with values $r(c_y) \in \{-1,0\}$, $r(sc_2) \in \{0,1\}$, $r(sc_3) \in \{0,1\}$, and $r(c_z) \in \{-1,0\}$ such that $r(sc_1) + r(c_y) = r(sc_2)$, $r(sc_2) + r(c_2) = r(sc_3)$, and $r(sc_3) + r(c_z) = r(sc_4)$. For all rows but one, there is a (unique) solution as indicated in the table.

$r(sc_1)$	$r(c_y)$	$r(sc_2)$	$r(c_2)$	$r(sc_3)$	$r(c_z)$	$r(sc_4)$
0	0	0	0	0	0	0
0			0			1
0	0	0	1	1	-1	0
0	0	0	1	1	0	1
1	-1	0	0	0	0	0
1	0	1	0	1	0	1
1	-1	0	1	1	-1	0
1	-1	0	1	1	0	1

The exception is when $r(sc_1) = 0$, $r(c_2) = 0$ and $r(sc_4) = 1$. However this situation cannot occur since this would entail that $r(c_1) = 0$, $r(c_2) = 0$ and $r(c_3) = 0$ (because $r(sc_4) = r(s_1) - r(c_3) = 1 - r(c_3)$), in contradiction with the assumption that f represents a solution of 3-*SAT*. Therefore, r defines a region of A that separates s_1 from s_2, as wanted. □

Inserting intermediate events c_y and c_z between the occurrences of the literals c_1, c_2, and c_3 in the transition system T_c that codes the clause $c = c_1 \vee c_2 \vee c_3$ is crucial to the proof of the above proposition. A similar technique will be employed in the next section.

2.2.2 The Elementary Net Synthesis Problem is NP-Complete

The reduction of 3-SAT to the state separation problem given by Hiraishi is elegant and simple, but it has no direct implication on the complexity of the elementary net synthesis problem. An independent reduction of 3-SAT to the (basic) elementary net synthesis problem for loop-free and simple initialized transition systems was proposed in [2], showing that this problem is NP-complete. We present below the principles of this reduction without providing the long and tedious proofs of the results.

It is first shown in [2] that 3-SAT is polynomialy equivalent to another problem of satisfaction of sets of clauses on the boolean ring $\mathbb{Z}/2\mathbb{Z}$. Recall that $\mathbb{Z}/2\mathbb{Z} = (\{0,1\}, +, 0, \cdot, 1)$ where $+$ is the sum modulo 2 (hence $z+z=0$ for all z) and $\cdot$ is the usual multiplication.

Definition 2.21 ([2]). *Let $X = \{x_0, \ldots, x_n\}$ be a set of boolean variables, with a distinguished element x_0. A* system of clauses over the boolean ring *is a pair (Σ, Π) where Σ is a finite set of* additive clauses *σ_α ($\alpha \in A$) and Π is a finite set of* multiplicative clauses *π_β ($\beta \in B$) with respective forms $x_{\alpha_0} + x_{\alpha_1} + x_{\alpha_2}$ and $x_{\beta_1} \cdot x_{\beta_2}$, subject to the following restrictions:*

1. *each additive clause has exactly three variables,*
2. *two additive clauses have at most one common variable,*
3. *each multiplicative clause has exactly two variables, and*
4. *the distinguished variable x_0 does not occur in any multiplicative clause.*

The system (Σ, Π) is said to be satisfiable *if there exists a boolean assignment for X such that $x_0 = 1$, $x_{\alpha_0} + x_{\alpha_1} + x_{\alpha_2} = 0$ for all $\alpha \in A$, and $x_{\beta_1} \cdot x_{\beta_2} = 0$ for all $\beta \in B$. Such boolean assignments are called* solutions *of (Σ, Π). Let CBR denote the problem whether a system of clauses (Σ, Π) has a solution.*

The problem is to code uniformly systems of clauses (Σ, Π) into loop-free and simple initialized transition systems $A(\Sigma, \Pi)$ with size polynomial in the size of (Σ, Π), such that (Σ, Π) has a solution if and only if $A(\Sigma, \Pi)$ is an elementary transition system. The intuition is to represent every additive clause by a directed cycle and every multiplicative clause by a diamond in $A(\Sigma, \Pi)$ in the sense given by the following definition.

Definition 2.22. *Let $A = (S, E, \delta, s_0)$ be an initialized transition system. A* directed cycle *(in A) is defined by a state $s \in S$ and a sequence of events $e_1 \ldots e_n \in E^*$ such that $\delta(s, e_1 \ldots e_n) = s$. A* diamond *(in A) is defined by a state s and two events e and e' such that $\delta(s, ee')$ and $\delta(s, e'e)$ are both defined and equal.*

To explain the coding, consider any non-trivial region r of $A = (S, E, \delta, s_0)$. Given $r : E \to \{-1, 0, 1\}$, define $abs(r) : E \to \mathbb{Z}/2\mathbb{Z}$ by setting down $abs(r)(e) = r(e) \mod 2$. Then $abs(r)(e_1) + \ldots + abs(r)(e_n) = 0$ for every directed cycle $(s, e_1 \ldots e_n)$ and $abs(r)(e) \cdot abs(r)(e') = 0$ for every diamond

(s, e, e'). Unfortunately, one cannot apply naively this principle for coding the satisfaction problem (Σ, Π) into an initialized transition system $A = A(\Sigma, \Pi)$, such that every region $r \in R(A)$ induces a solution $abs(r)$ of (Σ, Π) and every solution of (Σ, Π) is the abstraction $abs(r)$ of some region $r \in R(A)$. If every additive clause $x_{\alpha_0} + x_{\alpha_1} + x_{\alpha_2}$ in Σ was coded into a cycle $(s, x_{\alpha_0} x_{\alpha_1} x_{\alpha_2})$, then the satisfiability of (Σ, Π) could not be reduced to the problem whether $A(\Sigma, \Pi)$ is elementary. Indeed, it could occur that for some clause α and for some $i \in \{0, 1, 2\}$, $x_{\alpha_i} = 0$ for all solutions of (Σ, Π), and it would be impossible in this case to separate any pair of states s' and s" such that $s' \xrightarrow{x_{\alpha_i}} s$" in $A(\Sigma, \Pi)$.

This first problem may be avoided by replacing every additive clause $x_{\alpha_0} + x_{\alpha_1} + x_{\alpha_2}$ by two additive clauses coded into cycles of $A(\Sigma, \Pi)$, let $x_{\alpha_0} + x_{\alpha_3} + x_{\alpha_1} + x_{\alpha_4} + x_{\alpha_2} + x_{\alpha_5}$ and $\alpha + x_{\alpha_3} + x_{\alpha_4} + x_{\alpha_5}$ where x_{α_3}, x_{α_4}, x_{α_5} and α are fresh auxiliary variables, plus one multiplicative clause coded into a diamond of $A(\Sigma, \Pi)$, let $x_0 \cdot \alpha$ where x_0 is the distinguished variable. As $x_0 = 1$ and $x_0 \cdot \alpha = 0$ and $\alpha + x_{\alpha_3} + x_{\alpha_4} + x_{\alpha_5} = 0$ entail $x_{\alpha_0} + x_{\alpha_3} + x_{\alpha_1} + x_{\alpha_4} + x_{\alpha_2} + x_{\alpha_5} = x_{\alpha_0} + x_{\alpha_1} + x_{\alpha_2}$ in $\mathbb{Z}/2\mathbb{Z}$, the solutions of (Σ, Π) are set in correspondence in this way with the regions r of $A(\Sigma, \Pi)$ such that $r(x_0) \neq 0$. Now, pairs of states s' and s" such that $s' \xrightarrow{x_{\alpha_i}} s$" for some $i \in \{0, 1, 2\}$ can always be separated by regions r of $A(\Sigma, \Pi)$ such that $r(x_0) = 0$, i.e. by regions *that do not represent solutions of* (Σ, Π).

As a first approximation, let $A(\Sigma, \Pi)$ be constructed from all *state disjoint* cycles and diamonds induced from the respective equations $x_{\alpha_0} + x_{\alpha_3} + x_{\alpha_1} + x_{\alpha_4} + x_{\alpha_2} + x_{\alpha_5} = 0$, $\alpha + x_{\alpha_3} + x_{\alpha_4} + x_{\alpha_5} = 0$, $x_0 \cdot \alpha = 0$, and $x_{\beta_1} \cdot x_{\beta_2} = 0$, by defining for each component a transition $s_0 \xrightarrow{s} s$ from the initial state s_0 of $A(\Sigma, \Pi)$ to the initial state s of the component. The transitions of $A(\Sigma, \Pi)$ are thus labelled either by variables in X (the variables x_{α_i} for $\alpha \in A$ and $i \in \{0, 1, 2\}$ and the variables x_{β_j} for $\beta \in B$ and $j \in \{1, 2\}$), or with auxiliary variables α or x_{α_3}, x_{α_4}, x_{α_5} ($\alpha \in A$, all distinct), or with state variables s that occur only once.

Separating states of different components of $A(\Sigma, \Pi)$ is now easy, but it remains problematic to construct regions of $A(\Sigma, \Pi)$ separating pairs of states of a fixed component. The difficulty lies in the fact that each variable $x_k \in X$ may appear as the label of some transition in arbitrarily many cycles or diamonds. A region of a fixed component (a cycle or a diamond) can therefore not always be extended to a region of the global system $A(\Sigma, \Pi)$, even though all cycles and diamonds have disjoint sets of states.

In order to alleviate this second difficulty, one may proceed as follows. Define for each variable $x_k \in X$ as many fresh events x_k^α or x_k^β as clauses σ_α or π_β in which it occurs. Replace every occurrence of x_k in $A(\Sigma, \Pi)$ by x_k^α or x_k^β according to the clause from which this occurrence has been generated. Finally, for every new event x_k^γ, define a transition from the initial state s_0 of $A(\Sigma, \Pi)$ to the initial state of the transition system UV_k^γ shown in Fig. 2.6, labelled with a new event u_k^γ. Note that *all transition systems* UV_k^γ *with the*

$$\begin{tikzcd}
{} \arrow[dr, squiggly] & & & & \\
& s_0^\alpha \arrow[rr, "x^\alpha_{\alpha_0}"] & & s_3^\alpha \arrow[dr, "x_{\alpha_3}"] & \\
s_5^\alpha \arrow[ur, "x_{\alpha_5}"] & & (S^\alpha) & & s_1^\alpha \arrow[dl, "x^\alpha_{\alpha_1}"] \\
& s_2^\alpha \arrow[ul, "x^\alpha_{\alpha_2}"] & & s_4^\alpha \arrow[ll, "x_{\alpha_4}"] &
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[dr, squiggly] & & \\
& t_0^\beta \arrow[r, "x^\beta_{\beta_1}"] \arrow[d, "x^\beta_{\beta_2}"'] & t_1^\beta \arrow[d, "x^\beta_{\beta_2}"] \\
& t_2^\beta \arrow[r, "x^\beta_{\beta_1}"'] & t_3^\beta
\end{tikzcd}
\quad (T^\beta)$$

$$\begin{tikzcd}
{} \arrow[dr, squiggly] & & \\
& w_0^\alpha \arrow[r, "\alpha"] \arrow[d, "x_0"'] & w_1^\alpha \arrow[d, "x_0"] \\
& w_2^\alpha \arrow[r, "\alpha"'] & w_7^\alpha
\end{tikzcd}
\quad (W_L^\alpha)$$

$$\begin{tikzcd}
{} \arrow[dr, squiggly] & & \\
& w_3^\alpha \arrow[r, "x_{\alpha_3}"] & w_4^\alpha \arrow[d, "x_{\alpha_4}"] \\
& w_6^\alpha \arrow[u, "\alpha"] & w_5^\alpha \arrow[l, "x_{\alpha_5}"]
\end{tikzcd}
\quad (W_R^\alpha)$$

$$\begin{tikzcd}
{} \arrow[dr, squiggly] & & & \\
& u_k^\gamma \arrow[r, "y_k^\gamma"] \arrow[d, "x_k^\gamma"'] & \mathcal{U}_k^\gamma \arrow[r, "z_k^\gamma"] & u_k \arrow[d, "x_k"] \\
& v_k^\gamma & \mathcal{V}_k^\gamma \arrow[l, "z_k^\gamma"'] & v_k \arrow[l, "y_k^\gamma"']
\end{tikzcd}
\quad (UV_k^\gamma)$$

Fig. 2.6. the components of $A(\Sigma, \Pi)$

same index k share the states u_k, v_k and the transition $u_k \xrightarrow{x_k} v_k$. The role of the components UV_k^γ is explained by the following table, that displays all possible values $r(x_k)$, $r(x_k^\gamma)$, $r(y_k^\gamma)$ and $r(z_k^\gamma)$ for a region r of UV_k^γ.

$r(y_k^\gamma)$	$r(z_k^\gamma)$	$r(x_k^\gamma)$	$r(x_k)$
+1	-1	0	0
+1	0	+1	-1
-1	+1	0	0
-1	0	-1	+1
0	+1	+1	-1
0	-1	-1	+1
0	0	+1	+1
0	0	-1	-1
0	0	0	0

Let us explain how this table has been filled. Since consecutive transitions labelled y_k^γ and z_k^γ appear always in this order in UVk^γ, $r(y_k^\gamma) = +1 \Rightarrow r(z_k^\gamma) \in \{-1, 0\}$ and $r(y_k^\gamma) = -1 \Rightarrow r(z_k^\gamma) \in \{0, +1\}$. Similarly, $r(z_k^\gamma) = +1 \Rightarrow r(y_k^\gamma) \in \{-1, 0\}$ and $r(z_k^\gamma) = -1 \Rightarrow r(y_k^\gamma) \in \{0, +1\}$. The corresponding situations are enumerated in the leftmost columns of the upper part of the table. The entries of the rightmost columns are then uniquely determined, as readily verified. The remaining cases are when $r(y_k^\gamma) = r(z_k^\gamma) = 0$, and the only contraint imposed by the structure of UVk^γ is then $r(x_k^\gamma) = r(x_k)$.

In view of this table, $abs(r)(x_k^\gamma) = abs(r)(x_k) = abs(r)(x_k^\eta)$ in $\mathbb{Z}/2\mathbb{Z}$ for any two aliases x_k^γ and x_k^η of the same variable x_k used in different clauses

$\gamma, \eta \in \Sigma \cup \Pi$. However, at the same time, one may have either $r(x_k^\gamma) = r(x_k^\eta)$ or $r(x_k^\gamma) = -r(x_k^\eta)$. *This degree of freedom is crucial for producing regions* r *of* $A(\Sigma, \Pi)$ *from solutions* $f : X \to \{0,1\}$ of (Σ, Π) *such that* $f(x) = abs(r)(x)$ *for all* $x \in X$.

Altogether, $A(\Sigma, \Pi)$ is the assembly of all components shown in Fig. 2.6, namely S^α, W_L^α, and W_R^α (for every $\alpha \in A$), T^β (for every $\beta \in B$), and UV_k^γ (for every additive or multiplicative clause γ and for every variable x_k occurring in this clause). Every component is connected to the initial state s_0 of $A(\Sigma, \Pi)$ by a transition $s_0 \xrightarrow{s} s$ leading to its initial state s. All components UV_k^γ with the same index k share the transition $u_k \xrightarrow{x_k} v_k$.

Proposition 2.23. *If* $A(\Sigma, \Pi)$ *is an elementary transition system, then* (Σ, Π) *is satisfiable.*

Proof. If $A = A(\Sigma, \Pi)$ is an elementary transition system, then there must exist a region $r \in R(A)$ separating states w_0^α and w_2^α, hence necessarily $abs(r)(x_0) = 1$. As a result, $abs(r)(\alpha) = 0$ and $abs(r)(x_3^\alpha) + abs(r)(x_4^\alpha) + abs(r)(x_5^\alpha) = 0$ for all α. The restriction of the map $abs(r) : X \to \mathbb{Z}/2\mathbb{Z}$ on the set of original variables X of (Σ, Π) is a solution of this system, because $abs(r)(x_k^\gamma) = abs(r)(x_k)$ for all variables x_k and for all exponents γ. □

The converse proposition, stated below and established in [2], shows the NP-completeness of the elementary net synthesis problem.

Proposition 2.24. *If* (Σ, Π) *is satisfiable, then* $A(\Sigma, \Pi)$ *is an elementary transition system.* □

2.3 Algorithms of Elementary Net Synthesis

A variety of net synthesis problems have been dealt with and have received theoretical solutions in Chapter 1. In this section, we will design corresponding synthesis algorithms. We want algorithms that may synthesize both elementary nets or quasi-elementary nets from given transition systems or languages. We want to cover both exact net realization problems and approximate net realization problems.

In order to obtain optimal solutions to approximate net realization problems, there is no other way than to enumerate all regions of the given transition system or language, or all regions of a sufficiently complete subset thereof, e.g., the minimal regions. In order to obtain solutions to exact net realization problems, one can proceed differently. It has been shown in Chapter 1 that solving exact net realization problems amounts always to the following: given an initialized transition system A (or $\mathcal{U}(A)$ or L), search the set of regions $R(A)$ for an admissible subset w.r.t. a selection of the axioms SSP (state separation), ESSP (event-state separation), EESSP (strong event-state separation), EE (event effectiveness), and ES (event simpleness). For reasons explained later, one may disregard event effectiveness and focus on the *separation axioms* SSP, ESSP, SESSP, and ES (renamed *event separation* from now on).

An exact net realization problem may therefore be specified equivalently as a pair (A, γ) where A is an initialized transition system and γ is the set of *separation problems*, i.e., pairs $\{s, s'\}$ or $\{s, e\}$ or $\{e, e'\}$ where s, s' are states and e, e' are events, for which one wants to find separating regions. The set γ defines the *goal* of the search for admissible regions $R \subseteq R(A)$. If the search has been successful, then the net system $SN_R(A)$ synthesized from R may be returned as a solution to the net realization problem.

We know from Sect. 2.2 that each separation problem $\{s, s'\}$ or $\{s, e\}$ is NP-complete, and this complexity class pertains also to the separation problem $\{e, e'\}$, as the instances $\{e, e'\}$ where e and e' occur exactly once in A may be reduced to the separation problem $\{s, s'\}$. Therefore, one cannot expect to compute some region r separating a given pair $\{s, s'\}$ or $\{s, e\}$ or $\{e, e'\}$ without exploring to some extent the set $R(A)$, or the set $Rmin(A)$ (minimal regions are sufficiently complete for all separation axioms, see Chapter 1). However, in order to obtain reasonable synthesis algorithms, we set the following requirements on the process used to explore regions. First, the exploration should be goal oriented, i.e., it should not produce regions that do not solve any separation problem in γ. Second, one should produce as few non-minimal regions as possible. Third, one should not produce twice the same region.

In the rest of the section, $A = (S, E, \delta, s_0)$ is a fixed initialized transition system.

In order to enable a goal oriented exploration of the set of regions meeting the above requirements, we consider an abstraction of regions, called *rough regions* (Sect. 2.3.1). A rough region, as the name suggests, represents a (possibly empty) set of regions with constrained values $r(s)$ and $r(e)$ for a subset of states s and events e. Exploring $R(A)$ can be done by progressive refinements of rough regions, i.e., by progressive completions of their signature until some region is completely defined, or it may be recognized from the incomplete signature that at least one minimal region is compatible with this signature, or the rough region is found incoherent. We design in Sect. 2.3.2 an algorithm that generates rough regions and regions independently of any goal. This algorithm computes all minimal regions of a given transition system A, plus possibly some non minimal regions, and it is intended to be used in all forms of the approximate net realization problem. In Sect. 2.3.2, this general algorithm is adapted into a goal oriented algorithm, that may be used to decide on the feasability of exact net realization problems specified in the form (A, γ). We finally give in Sect. 2.3.2 some comments on the heuristics used in the synthesis tool PETRIFY [13].

2.3.1 Rough Regions

In this section, we introduce rough regions which are an abstraction of regions. Rough regions are a hybrid concept, in between regions and rough sets (see

[29] for the background). First, we recall the definition of rough sets. Next, we define the rough regions of an initialized transition system as rough sets of states satisfying regional conditions inspired from the conditions stated for regions in Def. 1.20. Finally, we show how one can compute rough regions from arbitrary rough sets by repeated inferences based on these conditions.

By now, the reader has probably experimented with the search for specific regions in a transition system. He may have observed an analogy with the game of Go: one puts a black stone on a point (a given state of the transition system) when one wants to include this state in the region. On the contrary, one puts a white stone when one wants that the considered state does not belong to the region. Points left unoccupied indicate states whose membership to the region is not yet determined.

Example 2.25. Suppose that one searches for a region r separating event a from state s in A, i.e., r is a solution to the separation problem $\{s, a\}$. If such a region exists, then r should contain the set $X_\bullet$ of all states enabling event a, and r should not intersect the set $X_\circ$ comprised of the considered state s and all states reached by a-labelled transitions (see Fig. 2.7). In other words, r should be compatible with the rough set $X = \langle X_\bullet, X_\circ \rangle$ according to Def. 2.26 stated hereafter. □

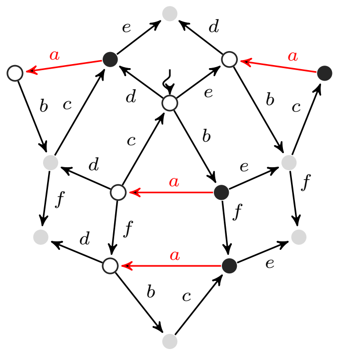


Fig. 2.7. rough set $X = \langle X_\bullet, X_\circ \rangle$ for the separation problem $\{s, a\}$ where s is the initial state

Definition 2.26. *A* rough set *(of states) X is a pair $\langle X_\bullet, X_\circ \rangle$ of subsets $X_\bullet, X_\circ \subseteq S$. The set $X_\bullet$ is the* positive *part of X, and $X_\circ$ is its* negative *part. The* extent *of the rough set $X = \langle X_\bullet, X_\circ \rangle$ is the set $[\![X]\!] = \{Y \subseteq S \mid X_\bullet \subseteq Y \subseteq S \setminus X_\circ\}$ of subsets of S* compatible *with X, i.e., containing the positive part of X and not intersecting its negative part. X is said to be* coherent *if its extent is not empty ($[\![X]\!] \neq \emptyset \Leftrightarrow X_\bullet \cap X_\circ = \emptyset$). A rough set is said to be* crisp *if $X_\circ = S \setminus X_\bullet$, in which case $[\![X]\!] = \{X_\bullet\}$. We allow* *ourselves to identify a subset $Y \subseteq S$ with the (crisp) rough set $\langle Y, S \setminus Y \rangle$.*

♢

When a rough set $X = \langle X_\bullet, X_\circ \rangle$ is coherent, its extent is the (non-empty) interval of sets $[X_\bullet, S \setminus X_\circ]$. The minimal set $X_\bullet$ contains all and only states which are required to belong to every set compatible with X; the maximal set $S \setminus X_\circ$ contains all and only states which are not excluded to belong to sets compatible with X. For technical reasons, we cannot work exclusively with intervals of sets and need considering possibly incoherent rough sets. We shall often use for rough sets $\langle X_\bullet, X_\circ \rangle$ an equivalent but more practical notation, as follows.

Notation 2.27 *A rough set can be identified with a map $X : S \to \mathbb{S}$ where $\mathbb{S}$ is the finite lattice shown next. The correspondence is given by:*

$$X_\bullet = \{ s \in S \mid X(s) \geq 1 \}$$
$$X_\circ = \{ s \in S \mid X(s) \geq 0 \}$$

and in the converse direction by:

$$\begin{array}{ll} X(s) = 0 & \textit{if} \;\; s \in X_\circ \setminus X_\bullet \\ X(s) = 1 & \textit{if} \;\; s \in X_\bullet \setminus X_\circ \\ X(s) = ! & \textit{if} \;\; s \in X_\bullet \cap X_\circ \\ X(s) = ? & \textit{if} \;\; s \notin X_\bullet \cup X_\circ \end{array}$$

$$\mathbb{S} = \begin{tikzcd} & ! & \\ 0 \arrow[ur] & & 1 \arrow[ul] \\ & ? \arrow[ul] \arrow[ur] & \end{tikzcd}$$

♢

Thus, $X(s) \geq 1$ means that state s should belong to every set compatible with the rough set X, and $X(s) \geq 0$ means that state s cannot belong to any set compatible with the rough set X. The relation $X(s) =?$ means that no constraint of membership bears upon state s. In contrast, the relation $X(s) =!$ means that two contradictory constraints of membership bear upon state s, hence X is incoherent. As a matter of fact, a rough set X is incoherent (i.e., $X_\bullet \cap X_\circ \neq \emptyset$) if and only if $X(s) =!$ for some state $s \in S$. The increase of the positive or negative information conveyed by rough sets is captured in the following refinement relation.

Definition 2.28. *Given rough sets $X : S \to \mathbb{S}$ and $X' : S \to \mathbb{S}$, X'* refines *X (notation: $X \leq X'$) if $X(s) \leq X'(s)$ for all $s \in S$.* ♢

The refinement relation is an order relation, and it turns the set $S \to \mathbb{S}$ of all rough sets (on S) into a finite lattice. The crisp rough sets (i.e., sets) coincide with the *coherent* rough sets which are maximal w.r.t. the refinement relation.

Remark 2.29. For any rough sets X and X', the following relations hold:

- $X \leq X' \Leftrightarrow X_\bullet \subseteq X'_\bullet \ \wedge \ X_\circ \subseteq X'_\circ$
- $[\![X]\!] = \{ Y \subseteq S \mid X \leq Y \}$
- $X \leq X' \Leftrightarrow [\![X']\!] \subseteq [\![X]\!]$ □

Rough sets (of states) present interest in the context of this book in as far as they represent sets of regions. For convenience, let us fix the notation.

Notation 2.30 *Given a rough set $X = \langle X_\bullet, X_\circ \rangle$, let $R(X)$ denote the set of regions of A compatible with X, i.e., $r \in R(X)$ if $r \in R(A)$ and $X_\bullet \subseteq r$ and $X_\circ \subseteq S \setminus r$.* $\diamondsuit$

Remark 2.31. $R(X) = \emptyset$ whenever X is an incoherent rough set, but $R(X) = \emptyset$ may also occur when X is a coherent rough set, since a non-empty interval $[X_\bullet, S \setminus X_\circ]$ may possibly not contain any region of A. $\square$

Exploring the set of regions $R(X)$ as we shall do in Sections 2.3.2 and 2.3.3 should be carried out in a rational way. Given a rough set X, one should therefore, before considering any other refinement, try to infer from X all logical consequences for regions r compatible with X (if such regions exist) of the rules stated in Def. 1.20 upon the possible values of $r(s)$ and $r(e)$ (for $s \in S$ and $e \in E$). Investigating this issue is the goal pursued in the rest of the section. To give some intuition, we propose first two examples.

Example 2.32. Let $X = \langle X_\bullet, X_\circ \rangle$ be the rough set described in Exemple 2.25. For every region $r \in R(X)$ and for every transition $s \xrightarrow{\lambda} s'$, the following relations must hold:

$$\begin{array}{lcl} (s \in r \wedge s' \notin r) & \Leftrightarrow & r(\lambda) = -1 \\ (s \notin r \wedge s' \in r) & \Leftrightarrow & r(\lambda) = +1 \\ (s \in r \Leftrightarrow s' \in r) & \Leftrightarrow & r(\lambda) = 0 \end{array}$$

Using these relations, one can check that there is a unique way to refine the rough set X into a crisp rough set r which is a region (i.e., $r \in R(X)$). The only possible values for the $r(\lambda)$ are indeed $r(a) = -1$, $r(b) = +1$, $r(c) = 0$, $r(d) = +1$, $r(e) = 0$, and $r(f) = 0$. The resulting region r is represented as a (crisp) rough set on the left of Fig. 2.8 (compare with Fig. 2.7 to see that this rough set is a refinement of X). The place of the synthesized net system derived from the region r is represented on the right of Fig. 2.8. $\square$

Example 2.33. Consider the separation problem $\{s_1, c\}$ in the initialized transition system depicted on the left of Fig. 2.9. Solving this separation problem amounts to finding a region compatible with the rough set $X = \langle X_\bullet, X_\circ \rangle$ represented in the middle of Fig. 2.9. The set $X_\bullet$ (black nodes) is the set of states where c is enabled. The set $X_\circ$ (white nodes) is the union of $\{s_1\}$ and the set of states reached by c-labelled transitions. For any region r in $R(X)$, the following relations must hold:

$$\begin{array}{lcl} (s_4 \notin r \ \wedge \ s_4 \xrightarrow{d} s_6) & \Rightarrow & r(d) \in \{0, +1\} \\ (s_7 \xrightarrow{d} s_1 \ \wedge \ s_1 \notin r) & \Rightarrow & r(d) \in \{-1, 0\} \end{array}$$

Hence necessarily, $r(d) = 0$, entailing that $s_6 \notin r$ and $s_7 \notin r$. Therefore, $R(X)$ contains a single region r, which is represented as a (crisp) rough set on the right of Fig. 2.9. This region is the unique region separating c from s_1. $\square$

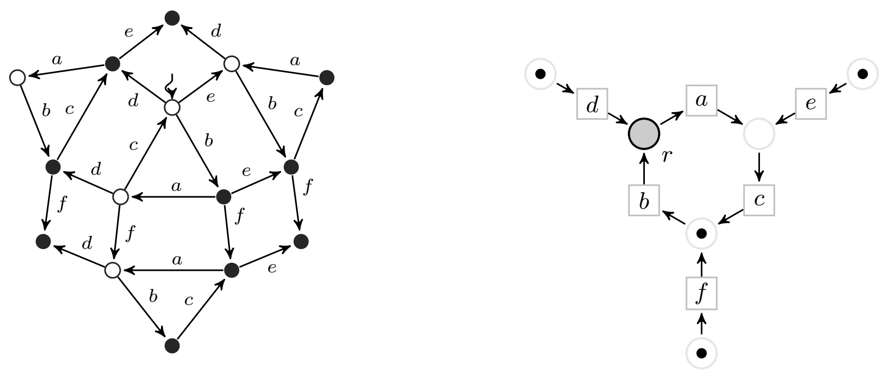


Fig. 2.8. a region $r \in R(X)$ represented as a crisp rough set and the corresponding place of the synthesized net system

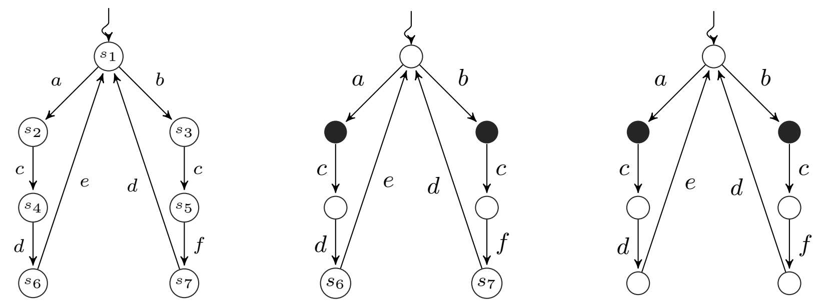


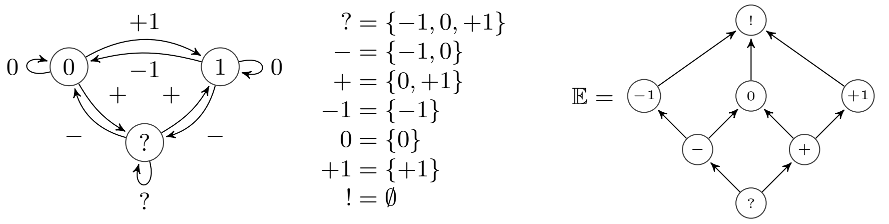
Fig. 2.9. computing a region for the separation problem $\{s_1, c\}$

In both examples above, the situation was simple because $R(X)$ contained one and exactly one region. In the general case, $R(X)$ may contain several regions, or no region at all. Moreover, in the general case, an iterative process is needed to infer all consequences of the rules stated in Def. 1.20 upon the possible values of $r(s)$ and $r(e)$ for an arbitrary region $r \in R(X)$.

In order to make the description of this inference process easy, we consider in a preliminary step a simplified objective as follows: given a coherent rough set X, infer the consequences of the constraints stated in Def. 1.20 on the possible values $r(e)$ for regions $r \in R(X)$ *without envisaging any feedback on the values* $r(s)$ *for states* $s \in X_? = \{s \in S \mid X(s) = ?\}$ (when X is a coherent rough set, $X(s) \in \{0, 1, ?\}$ for every state s and thus, $X_? = S \setminus (X_\bullet \cup X_\circ)$). This limited objective may be achieved by applying to every transition $s \xrightarrow{e} s'$ (of A) the following set of rules:

$$\begin{array}{lcl}
s \in X_\circ \ \wedge \ s' \in X_\bullet & \Rightarrow & r(e) = +1 \\
s \in X_\bullet \ \wedge \ s' \in X_\circ & \Rightarrow & r(e) = -1 \\
s \in X_\circ \ \wedge \ s' \in X_\circ & \Rightarrow & r(e) = 0 \\
s \in X_\bullet \ \wedge \ s' \in X_\bullet & \Rightarrow & r(e) = 0 \\
s \in X_\circ \ \wedge \ s' \in X_? & \Rightarrow & r(e) \in \{0, +1\} \\
s \in X_\bullet \ \wedge \ s' \in X_? & \Rightarrow & r(e) \in \{-1, 0\} \\
s \in X_? \ \wedge \ s' \in X_\circ & \Rightarrow & r(e) \in \{-1, 0\} \\
s \in X_? \ \wedge \ s' \in X_\bullet & \Rightarrow & r(e) \in \{0, +1\} \\
s \in X_? \ \wedge \ s' \in X_? & \Rightarrow & r(e) \in \{-1, 0, +1\}
\end{array} \tag{2.1}$$

All rules in 2.1 may be replaced equivalently with a single generic rule, which we now introduce. A few notations are needed. First, for any two states $q, q' \in \{0, 1, ?\}$, let $[q, q']$ be the label of the transition from q to q' in the transition system depicted on the left of Fig. 2.10, and let $[\![q, q']\!]$ be the corresponding subset of $\{-1, 0, +1\}$ given by the table in the middle of Fig. 2.10. Second,



$$\begin{aligned}
? &= \{-1, 0, +1\} \\
- &= \{-1, 0\} \\
+ &= \{0, +1\} \\
-1 &= \{-1\} \\
0 &= \{0\} \\
+1 &= \{+1\} \\
! &= \emptyset
\end{aligned}$$

Fig. 2.10. a lattice for classifying events w.r.t. a rough set

for $q \in \{0, 1, ?\}$, let $X_{\geq q} = \{s \in S \mid X(s) \geq q\}$ where $\geq$ is the order relation in the lattice $\mathbb{S}$ (Notation 2.27). When $X = \langle X_\bullet, X_\circ \rangle$ is a coherent rough set, $X_{\geq 0} = X_\circ$, $X_{\geq 1} = X_\bullet$, and $X_{\geq ?} = S$ (all states of A). Using these notations, for every transition $s \xrightarrow{e} s'$ (of A) the generic rule may be stated as:

$$(s \in X_{\geq q} \ \wedge \ s' \in X_{\geq q'}) \quad \Rightarrow \quad r(e) \in [\![q, q']\!]$$

For $q, q' \in \{0, 1, ?\}$, the sets $[\![q, q']\!]$, which are subsets of $\{-1, 0, +1\}$, form together with the empty set a lattice ordered by reverse set inclusion. This lattice $\mathbb{E}$ is shown on the right of Fig. 2.10, where $[\![q, q']\!]$ is represented as $[q, q'] \in \{?, -, +, -1, 0, +1\}$. The ordering in the lattice represents the increase of information: the least element '?' means the lack of information and the greatest element '!' means the excess of information, i.e., the presence of contradictory informations (which is excluded from coherent rough sets).

Given a coherent rough set X, by collecting for every event $e \in E$ all informations about the possible values of $r(e)$ obtained by applying the rules (2.1) to all transitions $s \xrightarrow{e} s'$, one gets a map $X : E \to \mathbb{E}$, called the *signature* of the rough set X, defined by

$$X(e) = \bigvee \left\{ [X(s), X(s')] \mid s \xrightarrow{e} s' \right\} = \bigcap \left\{ [\![X(s), X(s')]\!] \mid s \xrightarrow{e} s' \right\} \tag{2.2}$$

This definition of the signature of a rough set may be rephrased equivalently as follows.

Definition 2.34. *The* signature *of a rough set X is the least map $X : E \to \mathbb{E}$ (w.r.t. the pointwise ordering induced from the order relation in the lattice $\mathbb{E}$) such that the following relations hold for every transition $s \xrightarrow{e} s'$:*

$$\begin{array}{rcl}
(s \in X_\bullet \wedge s' \in X_\circ) & \Rightarrow & e \in X^\circ \\
(s \in X_\circ \wedge s' \in X_\bullet) & \Rightarrow & e \in {}^\circ X \\
(s \in X_\circ \wedge s' \in X_\circ) \vee (s \in X_\bullet \wedge s' \in X_\bullet) & \Rightarrow & e \in X^\perp \\
(s \in X_\bullet \vee s' \in X_\circ) & \Rightarrow & e \in X^- \\
(s \in X_\circ \vee s' \in X_\bullet) & \Rightarrow & e \in X^+
\end{array}$$

where X°, ${}^\circ X$, X^-, X^+ and $X^\perp$ are the following subsets of E:

$$\begin{array}{ll}
X^\circ & = \{e \in E \mid X(e) \geq -1\} \\
{}^\circ X & = \{e \in E \mid X(e) \geq +1\} \\
X^- & = \{e \in E \mid X(e) \geq -\} \\
X^+ & = \{e \in E \mid X(e) \geq +\} \\
X^\perp & = \{e \in E \mid X(e) \geq 0\}
\end{array}$$

♢

$X(e) =$	?	$-$	$+$	-1	0	$+1$	!
X°	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\notin$	$\in$
${}^\circ X$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$
X^-	$\notin$	$\in$	$\notin$	$\in$	$\in$	$\notin$	$\in$
X^+	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\in$	$\in$
$X^\perp$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$

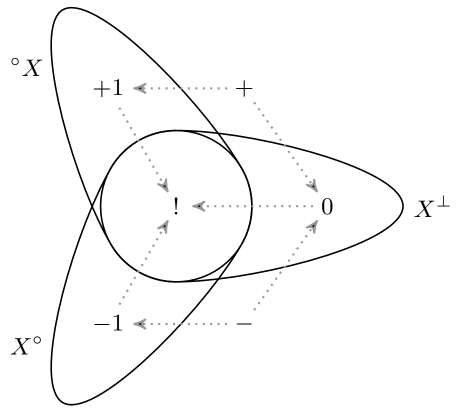


Fig. 2.11. membership of e to the sets $X^\circ, {}^\circ X, X^-, X^+, X^\perp$ of Def. 2.34

Remark 2.35. The signature of a crisp rough set (i.e., a set) takes its values $X(e)$ in $\{-1, 0, +1, !\}$. A crisp rough set X is a region if and only if $X(e) \neq !$ for every event e. In this case, $E = {}^\circ X \uplus X^\circ \uplus X^\perp$ and the place p of the synthesized net derived from the considered region has the flow relations given by ${}^\bullet p = {}^\circ X$ and $p^\bullet = X^\circ$. □

Remark 2.36. If $X \leq X'$ in the lattice $S \to \mathbb{S}$, then the signature of the rough set X is smaller than the signature of of the rough set X' in the lattice $E \to \mathbb{E}$. □

After this preliminary step, let us return to the general objective of inferring from X all logical consequences of the rules stated in Def. 1.20 upon the possible values of $r(s)$ and $r(e)$ for regions r compatible with X. We want to represent the result of the inference as a refinement of X, and more precisely as a rough region according to the following definition, which is reminiscent of Def. 1.20.

Definition 2.37. *A* rough region *is a rough set Y such that the following relations hold for all transitions $s \xrightarrow{e} s'$:*

$$\begin{array}{lll} e \in Y^{\circ} & \Rightarrow & (s \in Y_{\bullet} \ \wedge \ s' \in Y_{\circ}) \\ e \in {}^{\circ}Y & \Rightarrow & (s \in Y_{\circ} \ \wedge \ s' \in Y_{\bullet}) \\ e \in Y^{\perp} & \Rightarrow & [(s \in Y_{\bullet} \Leftrightarrow s' \in Y_{\bullet}) \ \wedge \ (s \in Y_{\circ} \Leftrightarrow s' \in Y_{\circ})] \end{array} \qquad \diamondsuit$$

The intuition under Def. 2.37 is the following: if all what we know about an unknown region $r \in R(Y)$ is that $Y_{\bullet} \subseteq r$ and $Y_{\circ} \cap r = \emptyset$, then one cannot increase this information by merely applying the rules stated in Def. 1.20, i.e., without predicting values $r(s)$ for any state $s \in S \setminus (Y_{\bullet} \cup Y_{\circ})$.

Remark 2.38. By Defs 1.20, 2.26 and 2.34, regions coincide with crisp rough regions, i.e., with rough regions which represent sets. □

We introduce now a transformation of rough sets directly inspired from Def. 2.37.

Definition 2.39. *For any rough set X, let $\alpha(X)$ be the least rough set Y larger than X such that the following relations hold for all transitions $s \xrightarrow{e} s'$:*

$$\begin{array}{lll} e \in X^{\circ} & \Rightarrow & (s \in Y_{\bullet} \ \wedge \ s' \in Y_{\circ}) \\ e \in {}^{\circ}X & \Rightarrow & (s \in Y_{\circ} \ \wedge \ s' \in Y_{\bullet}) \\ e \in X^{\perp} & \Rightarrow & [(s \in Y_{\bullet} \Leftrightarrow s' \in Y_{\bullet}) \ \wedge \ (s \in Y_{\circ} \Leftrightarrow s' \in Y_{\circ})] \end{array} \qquad \diamondsuit$$

When playing the Go-like game which was sketched in Exples 2.32 and 2.33, the principle followed is just to apply the transformation α iteratively to the given rough set X. The transformation α is an extensive and increasing operator in the finite lattice $S \to \mathbb{S}$, hence if one applies this transformation iteratively from X, one reaches sooner or later a fixpoint $\alpha^n(X) = \alpha^{n+1}(X)$. In view of the similarity between Definitions 2.37 and 2.39, the fixpoint $\alpha^n(X) = \alpha^{n+1}(X)$ is a rough region. We claim that is is indeed the least rough region larger than X. In order to establish this claim, we observe that for any two rough sets X and X', $X \leq X'$ entails $X(e) \leq X'(e)$ for all $e \in E$ (Remark 2.36), which entails in turn $X^{\circ} \subseteq X'^{\circ}$, ${}^{\circ}X \subseteq {}^{\circ}X'$, and $X^{\perp} \subseteq X'^{\perp}$. It follows, in view of the similarity between Definitions 2.37 and 2.39, that if a rough region Y is larger that $\alpha^i(X)$, then it must be larger than $\alpha^{i+1}(X)$, which establishes the claim by induction on i. The fixpoint $\alpha^n(X) = \alpha^{n+1}(X)$ is therefore equal to the rough region $\sigma(X)$ in the next definition.

Definition 2.40. *Given an initialized transition system $A = (S, E, \delta, s_0)$ and a rough set X (of states of A), the rough region $\sigma(X)$ induced by X is the least refinement Y of X such that the following relations hold for all transition $s \xrightarrow{e} s'$:*

$$\begin{array}{rl}
(s \in Y_\bullet \wedge s' \in Y_\circ) & \Leftrightarrow e \in Y^\circ \\
(s \in Y_\circ \wedge s' \in Y_\bullet) & \Leftrightarrow e \in {}^\circ Y \\
(s \in Y_\circ \wedge s' \in Y_\circ) \;\vee\; (s \in Y_\bullet \wedge s' \in Y_\bullet) & \Rightarrow e \in Y^\perp \\
(s \in Y_\circ \Leftrightarrow s' \in Y_\circ) \wedge (s \in Y_\bullet \Leftrightarrow s' \in Y_\bullet) & \Leftarrow e \in Y^\perp \\
(s \in Y_\bullet \vee s' \in Y_\circ) & \Rightarrow e \in Y^- \\
(s \in Y_\circ \vee s' \in Y_\bullet) & \Rightarrow e \in Y^+
\end{array} \quad \diamondsuit$$

The following proposition tells us that computing $\sigma(X)$ from X amounts as desired to extract from X all implicit informations on the values of $r(s)$ and $r(e)$ ($s \in S$ and $e \in E$) for an arbitrary region $r \in R(X)$.

Proposition 2.41. *For any rough set X, $R(\sigma(X)) = R(X)$.*

Proof. As $X \leq \sigma(X)$, $R(\sigma(X)) \subseteq R(X)$ by Rem. 2.29. To show the converse inclusion, let $r \in R(X)$. By Rem. 2.38, r coincides with a crisp rough region Y that refines X, i.e., $X \leq Y$. As $\sigma(X)$ is the least rough region larger than X, necessarily $\sigma(X) \leq Y$. By Rem. 2.38, $r \in R(\sigma(X))$, hence $R(X) \subseteq R(\sigma(X))$. $\square$

The proposition below shows that when computing $\sigma(X)$ by fixpoint iteration, one can stop the iteration at any step i such that $\alpha^i(X)(s) =!$ or $\alpha^i(X)(e) =!$ for some $s \in S$ or $e \in E$. In this case, as $\alpha^i(X) \leq \sigma(X)$, it follows indeed form the proposition that $\sigma(X)(s) =!$ and $\sigma(X)(e) =!$ for all $s \in S$ and $e \in E$, indicating that $R(X) = R(\sigma(X))$ is an empty set of regions.

Proposition 2.42. *Given an initialized transition system $A = (S, E, \delta, s_0)$, let $Y = \langle Y_\bullet, Y_\circ \rangle$ be any non-trivial* rough region *of A, i.e., $Y_\bullet \cup Y_\circ \neq \emptyset$. Then Y is coherent, i.e., $Y_\bullet \cap Y_\circ = \emptyset$, if and only if the sets ${}^\circ Y$, Y°, and $Y^\perp$ are pairwise disjoint. Moreover, A has a unique incoherent rough region Y, that satisfies $Y_\bullet = Y_\circ = S$ and ${}^\circ Y = Y^\circ = Y^+ = Y^- = E$, i.e., $Y(s) =!$ for every state s and $Y(e) =!$ for every event e.*

Proof. We show first that, if $s \in Y_\bullet \cap Y_\circ$, then for any transition $s \xrightarrow{e} s'$, $s' \in Y_\bullet \cap Y_\circ$. By Def. 2.34, $e \in Y^-$ because $s \in Y_\bullet$, and $e \in Y^+$ because $s \in Y_\circ$, hence $e \in Y^\perp$. By Def. 2.37, $s' \in Y_\bullet \cap Y_\circ$ since Y is a rough region. A symmetric reasoning may be used to show that, if $s' \in Y_\bullet \cap Y_\circ$, then for any transition $s \xrightarrow{e} s'$, $s \in Y_\bullet \cap Y_\circ$. As the underlying graph of A is connected (every state may be reached from the initial state) and every neighbour state of an incoherent state is incoherent, it follows by induction that $Y_\bullet = Y_\circ = S$, hence $\langle S, S \rangle$ is the unique incoherent rough region of A. As every event labels at least one transition, it follows from Def. 2.34 that necessarily, ${}^\circ Y = Y^\circ = Y^\perp = Y^+ = Y^- = E$. We have proved by the way that any rough region Y for which ${}^\circ Y$, Y°, and $Y^\perp$ are pairwise disjoint is coherent. To complete the proof of the proposition, it remains to show that the disjointness of subsets $^\circ Y$, Y°, and $Y^\perp$ is a necessary condition for the coherence of Y. Assume for instance that $e \in {}^\circ Y \cap Y^\perp$. As $e \in {}^\circ Y$, by Def. 2.34, there exists some transition $s \xrightarrow{e} s'$ with $s \in Y_\circ$ and $s' \in Y_\bullet$. As $e \in Y^\perp$, it follows by Def. 2.37 that $s' \in Y_\circ$ and $s \in Y_\bullet$, hence Y is incoherent. The case where $e \in {}^\circ Y \cap Y^\perp$ is similar. Finally, if one assumes $e \in {}^\circ Y \cap Y^\circ$, as $Y^\circ \subseteq Y^-$ and $^\circ Y \subseteq Y^+$, then necessarily $e \in Y^+ \cap Y^- = Y^\perp$ and the same reasoning as in the former two cases can be applied. □

We finally propose an example to illustrate the case where $R(Y)$ may be an empty set of regions although Y is a coherent rough region.

Example 2.43. In Section 2.2.1 we presented Hiraishi's reduction of 3-*SAT* to the state separation problem. The satisfiability of a clausal system was reduced to the separation problem $\{s_1, s_2\}$ for two specific states s_1 and s_2 of a transition system derived from the clausal system (see Fig. 2.5, p. 78). The separation problem $\{s_1, s_2\}$ may be equivalently specified by the rough set Y described in Fig. 2.12, i.e., the states s_1 and s_2 can be separated by a region if and only if $R(Y)$ is not empty. Therefore, deciding whether $R(Y)$ is empty is an NP-complete problem. This remark applies to rough sets and to rough regions as well, because the rough set Y described in Fig. 2.12 is in fact a (coherent) rough region. The signature computed from Eq. 2.2 is given by:

$$\begin{aligned}
&Y(v) = Y(\neg v) = + \\
&Y(c_1) = Y(c_3) = + \\
&Y(c_2) = ? \\
&Y(c_y) = Y(nc_y) = - \\
&Y(c_z) = Y(nc_z) = -
\end{aligned}$$

As a consequence, in view of Def. 2.34, $Y^\circ = {}^\circ Y = Y^\perp = \emptyset$, $Y^+ = \{v, \neg v, c_1, c_3\}$, and $Y^- = \{c_y, nc_y, c_z, nc_z\}$. As Y°, $^\circ Y$ and $Y^\perp$ are all empty, $\alpha(Y) = Y$ hence Y is a rough region. □

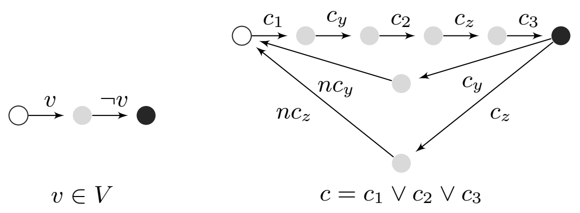


Fig. 2.12. rough set representation of a system of clauses

Remark 2.44. In Example 2.43, the sequence $c_1 c_y c_2 c_z c_3$ leads from the state s_2 to the state s_1. However, the Y-image $+ - ? - +$ of this sequence does not lead from state 0 to state 1 in the classifying transition system shown on the left of Fig. 2.10, even though $0 = Y(s_2)$ and $1 = Y(s_1)$. □

Example 2.43 has shown that the set of regions $R(X) = R(\sigma(X))$ represented by a rough set X may be empty even though the rough region $\sigma(X)$ induced by X is coherent. In Exercice 2.3, we show that for any rough set X, there exists a largest refinement $\rho(X)$ of X such that $R(X) = R(\rho(X))$, which entails that $R(X) = \emptyset$ if and only if $\rho(X)$ is incoherent. As the emptyness of $R(X)$ is an NP-complete problem, computing $\rho(X)$ must therefore require exponential time in the worst case, whereas computing $\sigma(X)$ takes polynomial time (since it amounts to a fixpoint computation in a finite lattice).

2.3.2 Extracting Regions from a Rough Region

Computing rough regions $\sigma(X)$ from rough sets $X = \langle X_\bullet, X_\circ \rangle$ serves to exhibit characteristics shared by all regions compatible with X, but it does not help exploring further the set of regions $R(X)$. In this section, we show how refining a rough region X into two rough regions X_1 and X_2, each of which is a refinement of X, such that $R(X) = R(X_1) \uplus R(X_2)$. In each of these two refinements one forces $r(e) = +1$ or $r(e) = -1$ or $r(e) = 0$ (for all regions $r \in R(X'))$ for one appropriate event $e \in E$. On this basis, one can construct from a rough region X a binary refinement tree with rough regions as internal nodes, selectors $e = +1$, or $e = -1$, or $e = 0$ as labels of arcs, and regions as leaves. By applying this construction to $2 \times |E|$ different rough sets, one obtains a global search tree for the whole set of regions $R(A)$. We propose an optimization of this global construction, aiming at producing all minimal regions ($Rmin(A)$) and as few non-minimal regions as possible, and avoiding that the same region can appear at two different leaves. The resulting set of regions R induces a net system $SN_R(A)$ close to $SN_{Rmin(A)}(A)$, and with a reachability graph isomorphic to the reachability graph of the saturated net system $SN(A)$, hence it affords a relatively small and qualitatively optimal solution to (all forms of) the approximate net synthesis problem.

To give an intuition, we present first a very simple example in which the reader will recognize the elementary transition system for mutual exclusion.

Example 2.45. Consider the transition system shown in Fig. 2.13. In this transition system, the event c is not enabled in the initial state s_0. The separation problem $\{s_0, c\}$ may be represented equivalently by the rough set $X = \langle X_\bullet, X_\circ \rangle$ given by $X_\bullet = \{s_1, s_6\}$ and $X_\circ = \{s_0, s_3, s_7\}$, i.e., a region r separates c from s_0 *iff* $r \in R(X)$. Let $Y = \sigma(X)$ be the induced rough set. Computing $\alpha(X)$ according to Def. 2.39 yields $\alpha(X)_\bullet = \{s_1, s_6\}$ and $\alpha(X)_\circ = \{s_0, s_3, s_4, s_7\}$. There are three different reasons why s_4 has been added to the negative part of the rough set:

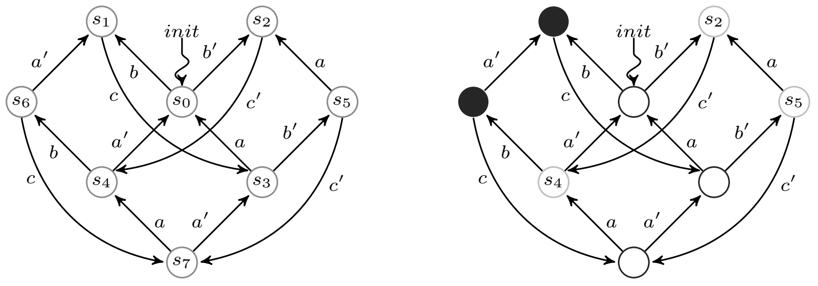
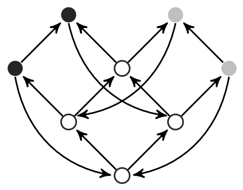


Fig. 2.13. rough set representation of the separation problem $\{s_0, c\}$

$$\begin{array}{lll} s_0 \in X_\circ & \wedge\, X(a') = 0 & \text{hence } a' \in X^\perp \\ s_7 \in X_\circ & \wedge\, X(a) = 0 & \text{hence } a \in X^\perp \\ s_6 \in X_\bullet & \wedge\, X(b) = +1 & \text{hence } b \in {}^\circ X \end{array}$$

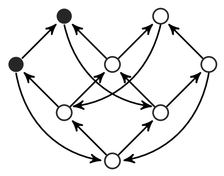
The reader can check that $\alpha^2(X) = \alpha(X)$, i.e., that no further refinement can be done without introducing new assumptions.

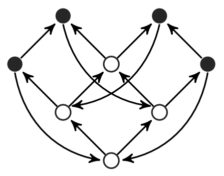


Thus $Y = \sigma(X)$, shown next, is given by $Y_\bullet = X_\bullet$ and $Y_\circ = X_\circ \cup \{s_4\}$ Note that $Y(b') = +$ and $Y(c') = -$. As $Y(b') = +$, for any $r \in R(Y)$, one must have $r(b') \in \{0, +1\}$. Therefore, one can split the set $R(Y)$ into $R_1 \uplus R_2$ where

$$R_1 = \{r \in R(Y) \mid r(b') = 0\} \quad \text{and} \quad R_2 = \{r \in R(Y) \mid r(b') = +1\}$$

Now, for any region r, $r(b') = 0$ implies $r(s_2) = r(s_5) = 0$ and $r(b') = +1$ implies $r(s_2) = r(s_5) = 1$. Therefore $R(X)$ contains contains exactly the following two regions (represented as crisp rough regions refining X):





□

Forcing regions $r \in R(X)$ to take a specific value $r(e) \in \{-1, 0, +1\}$, as was done in Example 2.45 is the object of the refinement operation introduced in the following definition.

Definition 2.46. *For any rough set $X = \langle X_\bullet, X_\circ \rangle$,*

1. *let $X[e := +1]$ be the rough set obtained by adding s to $X_\circ$ and s' to $X_\bullet$ for every transition $s \xrightarrow{e} s'$. Similarly,*
2. *let $X[e := -1]$ be the rough set obtained by adding s to $X_\bullet$ and s' to $X_\circ$ for every transition $s \xrightarrow{e} s'$. Finally,*

3. *let $X[e := 0]$ be the rough set obtained by adding s to $X_\bullet$ (resp. $X_\circ$) if s' belongs to $X_\bullet$ (resp. $X_\circ$), and conversely for s' and s, for every transition $s \xrightarrow{e} s'$.*

For $v \in \{-1, 0, 1\}$, let $X[e = v] = \sigma(X[e := v])$, i.e., the least rough region refining the rough set $X[e := v]$. In case when X is the totally undefined rough set ($X_\bullet = X_\circ = \emptyset$ and $R(X) = R(A)$), we abbreviate $X[e := v]$ and $X[e = v]$ to $[e := v]$ and $[e = v]$, respectively. $\diamondsuit$

Remark 2.47. For any rough set X and for any $v \in \{-1, 0, 1\}$, the following relations hold:

1. $R(X[e := v]) = R(X[e = v]) = \{r \in R(X) \mid r(e) = v\}$
2. $X(e) = - \quad \Rightarrow \quad R(X) = R(X[e = -1]) \uplus R(X[e = 0])$
3. $X(e) = + \quad \Rightarrow \quad R(X) = R(X[e = +1]) \uplus R(X[e = 0])$ $\square$

Let X be a coherent rough region which is neither crisp ($X_\bullet \cup X_\circ \neq S$) nor totally undefined ($X_\bullet \cup X_\circ = \emptyset$). Then, necessarily $X(e) \in \{-, +\}$ for some event e. By items (2) and (3) in Remark 2.47, one can always find two rough regions X_1 and X_2 which are refinements of X such that $R(X) = R(X_1) \uplus R(X_2)$. If moreover X has a non-empty positive part $X_\bullet$, i.e., the trivial region $\emptyset$ does not belong to $R(X)$, then the same property holds for the refinements X_1 and X_2 of X. By iterating the splitting procedure, one can construct from X a binary tree $T(X)$ providing a recursive decomposition of the set $R(X)$. The tree $T(X)$ has rough regions as nodes, and selectors $e = +1$ or $e = -1$ or $e = -0$ as labels of arcs.

From now on, we always suppose that X is a coherent rough region with a non-empty positive part $X_\bullet$. As we are mainly interested in minimal regions, we propose to optimize the construction of $T(X)$ in two ways. First, one discards any node Y which is an incoherent rough region (as it cannot contribute any region to $R(X)$). Second, one avoids to split any node Y such that Y is a coherent rough region and its positive part $Y_\bullet$ is a region. In the latter case, we say that Y is a *terminal* rough region. If Y is a terminal rough region then $Y_\bullet$ is the smallest region in $R(Y)$ and this region is non-empty (because $Y_\bullet \supseteq X_\bullet \neq \emptyset$). It is not worth exploring further $R(Y)$ in that case, since we are interested mainly in minimal regions and all regions in $R(Y)$ are larger than $Y_\bullet$. Note that a node Y of $T(X)$ which is a crisp rough region is always a terminal rough region (but not the other way round).

With these optimizations, it just remains to collect from $T(X)$ the set $R'(X)$ of all regions $Y_\bullet$ given by terminal nodes Y of this tree. Thus, $R'(X) = \{X_\bullet\}$ if X is terminal, $R'(X) = \emptyset$ if X is incoherent, and otherwise $R'(X) = R'(X_1) \cup R'(X_2)$ where X_1 and X_2 are the two sons of X, namely, $X_1 = X[e = +1]$ and $X_2 = X[e = 0]$ for some event e such that $X(e) = +$, or $X_1 = X[e = +1]$ and $X_2 = X[e = 0]$ for some event e such that $X(e) = -$.

Let $\min R'(X)$ denote the set of regions $r \in R'(X)$ which are minimal w.r.t. set inclusion in $R'(X)$. It is important to note that $\min R'(X)$ *does not depend* on the selection of the events e chosen for refining the nodes of

$T(X)$. Indeed, by construction, $\min R(X) \subseteq R'(X)$ and $R'(X) \subseteq R(X)$, hence $\min R'(X) = \min R(X)$. The remark below suggests that one can modify the algorithm which we have presented so as to compute directly the set $\min R(X)$.

Remark 2.48. We have seen that if X is a coherent rough region which is neither crisp nor totally undefined, then necessarily, $X(e) \in \{-, +\}$ for some event e. If $X(e) = -$, then there must exist some transition $s_1 \xrightarrow{e} s_1'$ with $s_1 \in X_\bullet$ and $s_1' \in X_?$ or some transition $s_2 \xrightarrow{e} s_2'$ with $s_2 \in X_?$ and $s_2' \in X_\circ$. If both types of transitions exist, then any two regions chosen from the respective sets $R(X[e = -1])$ and $R(X[e = 0])$ must be incomparable for set inclusion. Therefore in this case

$$\min R(X) = \min R(X[e = -1]) \uplus \min R(X[e = 0])$$

If $X(e) = +$, then a similar reasoning can be applied. If, at each stage where one should refine a node Y of $T(X)$, one could choose some event e with two types of transitions among the events satisfying $Y(e) \in \{-, +\}$, then one would get in the end $R'(X) = \min R(X)$. Problem 2.7 on page 107 shows how the algorithm which we have described may be adapted, based on this remark, to compute incrementally the set of all regions which are minimal in $R(X)$ (assuming that $X_\bullet \neq \emptyset$, hence that $\emptyset \notin R(X)$). □

It is worth noting that regions $r \in \min R'(X)$ need not be minimal regions of A. In particular, regions $Y_\bullet$ defined by terminal nodes $Y = \langle Y_\bullet, Y_\circ \rangle$ of $T(X)$ may contain strictly smaller non-empty regions of A *which do not belong to* $R(X)$. There is however a crucial case where $\min R(X) = Rmin(A) \cap R(X)$, identified by the following proposition.

Proposition 2.49. *For any rough region X of the form $[e = -1]$, or $[e = +1]$ $\min R(X) = Rmin(A) \cap R(X)$.*

Proof. First, note that every rough region X of the form $[e = -1]$, or $[e = +1]$ has a non-empty positive part. We have to show that every non-empty region $r \in \min R(X)$ is a minimal region of A. Suppose for a contradiction that $r' \subset r$ for some non-empty region r' of A. Let $r'' = r \setminus r'$. If X equals $[e = -1]$ (resp. $[e = +1]$), then either $r'(e) = -1$ (resp. $r'(e) = +1$) and then $r' \in R(X)$, or $r''(e) = -1$ (resp. $r''(e) = +1$), and then $r'' \in R(X)$. Therefore, r was not minimal in $R(X)$, showing the contradiction. □

Prop. 2.49 tells us that, if we gather all individual trees $T[e = -1]$ or $T[e = +1]$ into a single tree T, with corresponding selectors $e = -1$ or $e = +1$ labelling the initial arcs, then all minimal regions of A appear as terminal nodes of this tree. This gives an algorithm for enumerating all minimal regions, plus some non-minimal regions, but we are not done because this enumeration is not necessarily free from repetitions, i.e., several terminal nodes may contribute the same region. This may be easily corrected in view of the remark below.

Remark 2.50. If $E = \{e_1, \ldots, e_n\}$ is an enumeration of the set of events of the initialized transition system A, then

$$R(A) \setminus \{\emptyset\} = \biguplus_{1 \le i \le n} \{r \in R([e_i = -1]) \mid \forall 1 \le j < i \quad r(e_j) \neq -1\}$$
$$\uplus \biguplus_{1 \le i \le n} \left\{ r \in R([e_i = +1]) \,\middle|\, \begin{array}{l} \forall 1 \le j \le n \;\; r(e_j) \neq -1 \\ \forall 1 \le j < i \;\; r(e_j) \neq +1 \end{array} \right\}$$

□

In order to compute a "small" set of regions $R'(A)$ including all minimal regions of A, i.e., such that $\min(R(A) \setminus \{\emptyset\}) \subseteq R'(A) \subseteq R(A)$, one can proceed as follows. First compute for all events e_i the rough regions X_i and X_{n+i} defined as $[e_i = -1]$ and $[e_i = +1]$, respectively. For $k = 1$ to $2n$, if $X_k(e_j) = -1$ and $j < k$ or $X_k(e_j) = +1$ and $n + j < k$ for some $j \le n$, then let X_k be redefined as the incoherent rough region. Next, for $k = 1$ to $2n$, construct from the rough region X_k the corresponding tree $T(X_k)$, but whenever a new node Y is generated in this tree by the refinement process, if $Y(e_j) = -1$ and $j < k$ or $Y(e_j) = +1$ and $n + j < k$ for some $j \le n$, then let Y be redefined as the incoherent rough region. Let $R'(X_k)$ be the collection of regions $Y_\bullet$ defined by the terminal nodes $Y = \langle Y_\bullet, Y_\circ \rangle$ of the tree $T(X_k)$ thus obtained, and let $R'(A) = \cup_{k \le 2n} R'(X_k)$. By Remark 2.50, the sets $R'(X_k)$ are pairwise disjoint, and $\min(R(A) \setminus \{\emptyset\}) = \min R'(A)$ as desired.

Remark 2.51. Problem 2.8 on page 110 shows that one can proceed in a similar way to compute incrementally the set $Rmin(A)$ of all *minimal* regions of A.

□

2.3.3 Net Synthesis Algorithms

In Chap. 1, we have introduced and studied different types of net synthesis problems, depending on whether a given initialized transition system A should be realized by a net system up to transition system isomorphism or up to language equivalence, on whether one wants to synthesize an arbitrary net system or an elementary net system, and on whether one wants this net system to be contact-free or not. For all types of problems, deciding upon the existence of an exact solution to the problem and synthesizing a corresponding net system reduces to finding in a transition system A (or $\mathcal{U}(A)$ or L) a subset of regions $R \subseteq R(A)$ that enforce a selection of the following axioms:

state separation: for all pairs of distinct states s and s'

$$\exists r \in R \qquad (s \in r \wedge s' \notin r) \vee (s \notin r \wedge s' \in r)$$

event-state separation: for all pairs of state s and event e disabled in s

$$\exists r \in R \qquad (r \in {}^\circ e \wedge s \notin r) \vee (r \in e^\circ \wedge s \in r)$$

strong event-state separation: for any event e disabled in state s

$$\exists r \in R \qquad r \in {}^\circ e \wedge s \notin r$$

event separation: for all pairs of distinct events e and e'

$$(R \cap {}^\circ e \neq R \cap {}^\circ(e')) \vee (R \cap e^\circ \neq R \cap (e')^\circ)$$

event effectiveness: $\forall e \in E \quad R \cap {}^\circ e \neq \emptyset$.

For more precision, let us recall some results:

1. An initialized transition system A is isomorphic to the reachability graph of a net system N if and only N is isomorphic to the net system $SN_R(A)$ synthesized from a subset of regions $R \subseteq R(A)$ enforcing state separation and event-state separation.
2. A loop-free and simple initialized transition system A is isomorphic to the reachability graph of an elementary net system N if and only N is isomorphic to the net system $SN_R(A)$ synthesized from a subset of regions $R \subseteq R(A)$ satisfying state separation and event-state separation.
3. An initialized transition system A is language equivalent to a net system N if and only N is isomorphic to the net system $SN_R(\mathcal{U}(A))$ synthesized from a subset of regions $R \subseteq R(\mathcal{U}(A))$ enforcing event-state separation, where $\mathcal{U}(A)$ is the limited unfolding of A.
4. An initialized transition system A is language equivalent to an elementary net system N if and only N is isomorphic to the net system $SN_R(\mathcal{U}(A))$ synthesized from a subset of regions $R \subseteq R(\mathcal{U}(A))$ enforcing event-state separation, event separation and event effectiveness in $\mathcal{U}(A)$, the limited unfolding of A.
5. In all cases above, similar results hold for contact-free nets up to replacing event-state separation with strong event-state separation. Moreover, minimal regions are sufficiently complete w.r.t. all axioms, and whenever event-state separation holds, strong event-state separation can be enforced by minimal regions.

In this section, we design a general net synthesis algorithm which applies to all types of synthesis problems, with one exception, and which produces contact-free nets exclusively (in view of the last result recalled above, this is always possible when the synthesis problem has a solution). The exception is the synthesis of *elementary* net systems from transition systems A up to language equivalence. This is in fact the unique form of the net synthesis problem in which the axiom of event effectiveness enters the game. However, for this problem, event-state separation is also required and if it holds, then event effectiveness holds if and only if no event e is enabled in every state of $\mathcal{U}(A)$, and this can be checked *a priori* on A. Therefore, the general net synthesis algorithm which we propose covers also the synthesis of elementary net systems up to language equivalence for initialized transition system A passing this test.

Definition 2.52. *Given an initialized transition system $A = (S, E, \delta, s_0)$, a* separation problem *is any pair $\{s, s'\}$ or $\{s, e\}$ or $\{e, e'\}$ where s, s' are states and e, e' are events, such that $s \neq s'$ or $\neg(s \xrightarrow{e})$ or $e \neq e'$, respectively. A separation problem is* feasible *if the considered pair of states / events is separated by some region $r \in R(A)$, called a solution to the separation problem. A* separation goal *γ is a set of separation problems. A separation goal is* feasible *if every separation problem in this goal is feasible. A set of regions is* admissible *w.r.t. goal γ if it supplies solutions to all separation problems in γ.* ♢

Deciding whether a net synthesis problem has feasible solutions reduces to deciding whether some corresponding separation goal is feasible. Given an initialized transition system A and a separation goal γ relative to A, we want to produce for γ an admissible set of regions $R \subseteq R(A)$ if this goal is feasible, thus yielding the net system $SN_R(A)$ as a solution to the net synthesis problem, or to produce in the converse case the residual subset of separation problems in γ which are not feasible. Moreover, we want to be free to impose or not to impose the constraint that $SN_R(A)$ should be contact free. We adapt for this purpose the techniques presented in Sect. 2.3.2.

We describe now the generic net synthesis algorithm which results from this adaptation. In the sequel, $A = (S, E, \delta, s_0)$, $\{e_1, \ldots, e_n\}$ is an enumeration of the set of events E, and $\gamma \subseteq ((S \times S) \cup (S \times E) \cup (E \times E))$ is a separation goal. For $i = 1$ to n, let X_i and X_{n+i} denote the rough regions $[e_i = -1]$ and $[e_i = +1]$, respectively (see Def. 2.46). The algorithm visits the trees $T(X_k)$ according to the increasing order on subscripts k, with the two optimizations described in the end of Sect. 2.3.2. The visit of each tree $T(X_k)$ is stopped as soon as it cannot help to solve further the residual separation goal. The exploration of the sequence of trees $T(X_k)$ as soon as the set of collected regions $\cup_{k \leq j} R'(X_k)$ is admissible for the given goal γ. To better explain the control of the iteration and the production of the output, let us give one more definition.

Definition 2.53. *Given a separation goal γ and a set of regions $R \subseteq R(A)$, let $\gamma^R \subseteq \gamma$ denote the set separation problems in γ solved by regions in R.*

♢

The net synthesis algorithm may then be described by the following pseudo-code:

```
$solve(\gamma) =$
begin
    $R \leftarrow \emptyset$
    for $k = 1$ to $2n$ do
        if $\gamma \neq \emptyset$ then
            $R_k \leftarrow visit(X_k, k, \gamma)$
            $R \leftarrow R \cup R_k$
            $\gamma \leftarrow \gamma \setminus \gamma^{R_k}$
    done
    if $\gamma = \emptyset$
        then  $return(R)$
        else  $return(R, \gamma)$
end

$visit(X, k, \gamma) =$
begin
    if  $X$ is incoherent or
        $X(e_j) = -1$ for some $j < k$ or
        $X(e_j) = +1$ for some $j < k - n$ or
        $X$ is terminal and $\gamma^R = \emptyset$ for $R = \{X_\bullet\}$
    then return($\emptyset$)
    if $X$ is terminal then return($\{X_\bullet\}$)
    otherwise choose some $e \in E$ such that $X(e) \in \{-, +\}$
        if $X(e) = +$ then $R_1 \leftarrow visit(\sigma(X[e := +1]), k, \gamma)$
                           $R_2 \leftarrow visit(\sigma(X[e := 0]), k, \gamma \setminus \gamma^{R_1})$
                           return($R_1 \cup R_2$)
        if $X(e) = -$ then $R_1 \leftarrow visit(\sigma(X[e := -1]), k, \gamma)$
                           $R_2 \leftarrow visit(\sigma(X[e := 0]), k, \gamma \setminus \gamma^{R_1})$
                           return($R_1 \cup R_2$)
end
```

Some comments follow. The algorithm returns either an admissible set of regions R for the given goal γ, hence a solution to the net synthesis problem, or a set of regions R and a residual goal comprised of all unfeasible separation problems within the given goal γ. In the latter case, the synthesis problem has no exact solution and the net synthesized from R is the best approximate solution. This is done by successive visits to the trees $T(X_k)$ for $k = 1$ to $2n$. Visiting $T(X_k)$ with a goal γ produces a set R_k of regions extracted from $R'(X_k)$, each of which solves some separation problem in γ, and a residual goal $\gamma \setminus \gamma^{R_k}$. The residual goal is comprised of all separation problems which have no solution in $R'(X_k)$, and defines the goal of the visit to the next tree. The parameter X of $visit(X, k, \gamma)$ is either the rough region X_k or a rough region which has been derived from X_k by refinement operations. The third parameter is the goal assigned to the visit of the subtree of $T(X_k)$ rooted at X. Each tree $T(X_k)$ is explored by depth first search and from left to right. Following the simplification rules indicated after Remark 2.50 p. 98, the subtree of $T(X_k)$ rooted at X is not really visited when it is sure that $R(X)$ has already been entirely explored, i.e. when $X(e_j) = -1$ for some $j < k$ or $X(e_j) = +1$ for some $j < k - n$. For a terminal node X, the result of $visit(X, k, \gamma)$ is the singleton set $\{X_\bullet\}$ if the region $X_\bullet$ solves some separation problem in γ, or the empty set otherwise. In case when two sibling nodes Y_1 and Y_2 are explored, the subgoal reached by Y_1 is subtracted from the goal assigned to Y_2. The sets of regions R_1 and R_2 produced at sibling nodes are collected and returned inductively.

A first improvement may be brought by avoiding to make an actual procedure call $visit(X, k, \gamma)$ at all times when it can be checked from $X_\bullet$, $X_\circ$, and the induced signature of X, that no region in $R(X)$ can solve any separation problem in γ.

A second and more significant improvement is suggested in the following remark, which is in the line of the earlier Remarks 2.48 and 2.51.

Remark 2.54. Problem 2.10 on page 112 shows how adapting the algorithm so that it computes an admissible set of *minimal* regions for the given goal γ, or decides that no such set exists, without producing twice any minimal region. □

The net synthesis algorithm which we have defined computes an admissible set R of regions (or minimal regions, with the above adaptation) for the given goal γ . However, R is not necessarily minimal w.r.t. set inclusion amongst admissible sets of regions. It is not difficult to extract a minimal admissible set of regions from a given admissible set of regions, e.g. by using the method presented in Prob. 1.8 (on page 60). This minimization procedure is however highly time consuming since it requires computing first for each region in R the exact list of separation problems in γ which this region solves, which our net synthesis algorithm precisely avoids by computing a residual goal after each step.

2.3.4 The Heuristic Approach of Petrify

When a net synthesis problem given by a transition system A and a goal γ has no solution, the synthesis algorithm which has been presented in Section 2.3.3 returns a list of unfeasible separation problems in the given goal γ and an approximate solution to the problem, given by a set of regions of A or by the net system $SN_R(A)$ synthesized from this set of regions. Using this feedback, one may try to modify A and iterate synthesis. No optimal restructuring technique has been developped yet for this purpose, but at least, heuristics have been proposed and implemented in the synthesis tool PETRIFY [13]. In this section, we present briefly the principles of PETRIFY and the heuristics used for label splitting, and then discuss this concept in a more general context.

Technically speaking, PETRIFY is dedicated to the synthesis of net systems up to a language preserving folding operation. From the results established in Sect. 1.6 and Sect. 1.7 of Chap. 1, we know that the problem of net synthesis up to language equivalence may be reduced to the former problem by a limited unfolding of the given transition system A.

Recall that $A = (S, E, \delta, q_0)$ may be realized up by an elementary net system up to a language preserving folding operation if and only if the axioms of event-state separation (or equivalently, strong event-state separation), event effectiveness, and event simpleness are satisfied in A. In this case, $A \rhd RG(SN_R(A))$ for any set of regions $R \subseteq R(A)$ which is admissible for the considered axioms. Moreover, the minimal regions of A are sufficiently complete for this problem.

In PETRIFY, event simpleness is disregarded, and one searches for strongly admissible sets of minimal regions w.r.t. the remaining two axioms, namely, event effectiveness and strong event-state separation. However, it may occur that $SN_R(A)$ is not an elementary net system, even though $R \subseteq Rmin(A)$ is an admissible set of regions for both axioms, simply because this net system has equivalent transitions. To keep working with elementary nets, we shall assume in this section that the construction of net systems $SN_R(A)$ from sets of regions R is slightly modified as follows: the transitions of $SN_R(A)$, which were till now the events of A, are henceforth the equivalence classes of events of A induced by the relation $e \equiv e'$ if $r(e) = r(e')$ for every $r \in R$.

Event-effectiveness requires that every event has some pre-region, i.e. that ${}^\circ e$ differs from the empty set for every e. Event-state separation requires that, whenever $\delta(s, e)$ is undefined for some $s \in S$, $s \notin r$ for some region $r \in {}^\circ e$. As minimal regions are sufficiently complete w.r.t. these axioms, the algorithms used in PETRIFY are focussed on minimal pre-regions of events.

In order to check event-state separation for a given event e, one computes first the *actual enabling set* $AES(e) = \{s \in S \mid \delta(s, e) \ defined\}$. Taking $AES(e)$ as a seed, one examines stepwise all extensions of this set liable to be minimal pre-regions of e or to be included in minimal pre-regions of e. One can then compute the *region enabled set* $RES(e) = \cap \{r \in Rmin(A) \mid r^\circ e\}$. If $AES(e) = RES(e)$, then event-state separation holds for e. In the converse case, no folding of A can be realized by an elementary net system.

However, when $AES(e)$ is strictly included in $RES(e)$ for some e, PETRIFY does not simply notify that A cannot be realized by an elementary net system. Instead of this, PETRIFY computes a relabelled version A' of A which may be realized by an elementary net system, and it produces a net system N' such that $A' \rhd RG(N')$. An algorithm is proposed to compute an initialized transition system A' as close as possible to the original A. This algorithm is based on the following heuristic.

Whenever $AES(e) \subset RES(e)$ for some e, compute for each set of states Q between these two bounds ($AES(e) \subseteq Q \subseteq RES(e)$) the number of violations of the border crossing conditions that would hold if Q was a region (e.g. for some e', some transitions labelled e' have their source and target in Q, and some other transitions labelled e' exit from Q). Choose Q with the least number of violations of the border crossing conditions, and as small as possible within this classs.0 Then *force* Q to become a region by relabelling A as follows. For each event e' violating the border crossing conditions w.r.t. the set Q, change labels e' of transitions entering, resp. exiting Q to e'_1, resp. to e'_2. Provided that the axiom of event effectiveness is satisfied in A, an iterative application of this relabelling procedure leads always to an elementary transition system A' as desired, since an initialized transition system in which every event occurs exactly once must be an elementary transition system.

This relabelling technique is known as *label splitting* or *event splitting*, since it consists in splitting, for some labels e, the set of transitions $s \xrightarrow{e} s'$ of A into several subsets, in each of which e is replaced by a new label. Suppose that e has been split into a set of labels $\{e_1, \ldots, e_n\}$, yielding a new transition system A'. Suppose that A' can be realized by a net system N' up to isomorphism of transition systems. Then $e_1, \ldots, e_n$ are transitions of N', and they appear as such in the reachability graph of N'. However, if one merges backwards all labels $e_1, \ldots, e_n$ into a single label e, then one retrieves A (up to an isomorphism). Realizing transition systems by net systems up to label splitting is indeed just the same as realizing transition systems by *labelled* net systems. Applying label splitting to a transition system A produces a transition system A' with more regions, and if it is applied until every label occurs exactly once, then all sets of states are regions and the net realization problem becomes trivial. The issue is to find an optimal strategy to guarantee the success of net synthesis at the lowest cost, where the cost may be e.g. the number of new labels used, or the number of the relabelled transitions. This is an important issue for the practical applications of net synthesis.

Problems

2.1. This exercice uses Propositions 2.1 (p. 64), 2.12 (p. 68), and 2.13 (p. 69)
(a) Show that the intersection $r_1 \cap r_2$ of two regions r_1 and r_2 is a region if and only if their union $r_1 \cup r_2$ is a region.
(b) Show that, for any transition system A, the subsets of regions of A which, when equipped with set theoretic union and complementation, may be seen as boolean algebras, are in bijective correspondence with the partitions of the set of states into regions.
(c) Consider the transition system introduced in Prob. 1.3 (on page 59). Draw a diagram showing the regions of this transition system ordered by set inclu-

sion. Find the maximal subsets of regions which define boolean algebras. What can you say about the regions in these subsets? Construct the corresponding components of the net system synthesized from all regions.

2.2 (From [2]). Show that 3-SAT reduces polynomially to the satisfiability problem for systems of clauses over the boolean ring (definition 2.21).

2.3. The *dual* of a rough set X is the rough set $\overline{X} = \langle \overline{X}_\bullet, \overline{X}_\circ \rangle$ defined by $\overline{X}_\bullet = X_\circ$ and $\overline{X}_\circ = X_\bullet$.
(a) Show that *(i)* $R(\overline{X}) = \{\overline{r} \mid r \in R(X)\}$, where $\overline{r}$ is the complement of r, *(ii)* for any coherent and non-trivial rough set X (i.e., $X_\bullet \cap X_\circ = \emptyset$ and $X_\bullet \cup X_\circ \neq \emptyset$) the sets $R(X)$ and $R(\overline{X})$ are disjoint (i.e., a region and its complement cannot both be compatible with X), and *(iii)* $X \leq Y \Leftrightarrow \overline{X} \leq \overline{Y}$
(b) Let the *completion* $Y = \rho(X)$ of a rough set $X = \langle X_\bullet, X_\circ \rangle$ be defined as the rough set Y with $Y_\bullet = \bigcap_{r \in R(X)} r$ and $Y_\circ = \bigcap_{r \in R(\overline{X})} r$. Observe that $\overline{\rho(X)} = \rho(\overline{X})$, $\rho(\langle \emptyset, \emptyset \rangle) = \langle \emptyset, \emptyset \rangle$, and $\rho(X) = \langle S, S \rangle$ for any incoherent rough set X. Prove that completion is a closure operation, i.e., that

1. $X \leq \rho(X)$
2. $X \leq Y \Rightarrow \rho(X) \leq \rho(Y)$
3. $\rho(\rho(X)) = \rho(X)$

Prove moreover that $R(\rho(X)) = R(X)$.
(c) Show that the completion of a rough set X is the largest refinement Y of X such that $R(X) = R(Y)$. More precisely, show that for any refinement Y of X, $R(X) = R(Y) \Leftrightarrow Y \leq \rho(X)$.
(d) Observe that $R(X) = \emptyset$ if and only if $\rho(X) = Y$ is the largest incoherent rough set, defined by $Y_\bullet = Y_\circ = S$. Show that computing the completion of a rough set is NP-Complete.

2.4. (a) Let transition systems $(S, E, \mathit{\Delta})$ be defined as in Def. 1.6 except that the initial state is missing. Observe that the definition of regions (Def. 1.20) does actually not depend on initial states, hence it may be applied also to transition systems. Show that r is region of the transition system $(S, E, \mathit{\Delta})$ if and only if

$$s \xrightarrow{e} s' \in \mathit{\Delta} \quad \Rightarrow \quad r(s) \xrightarrow{r(e)} r(s') \in \mathit{\Delta}_\tau$$

$$\begin{tikzcd} 0 \arrow[loop left, "0"] \arrow[r, bend left, "+1"] & 1 \arrow[l, bend left, "-1"] \arrow[loop right, "0"] \end{tikzcd}$$

where $\tau = (Q, L, \mathit{\Delta}_\tau)$ is the transition system depicted above, with sets of states $Q = \{0, 1\}$ and set of events $L = \{-1, 0, +1\}$. Observe that a similar property does not hold for rough regions when replacing this transition system with the one in Fig. 2.10 (see Remark 2.44).
(b) We have presented the lattice $\mathbb{E}$ (Fig. 2.10 on page 89) as the set of subsets of $\{-1, 0, +1\}$ ordered by reverse inclusion; similarly one can present the lattice $\mathbb{S}$ (see figure on page 86) as the set of subsets of $\{0, 1\}$ ordered by reverse inclusion, with $? = \{0, 1\}$, $0 = \{0\}$, $1 = \{1\}$, and $! = \emptyset$.

Show that the operation $[\cdot,\cdot]$ defined on page 89 is the restriction to $\mathbb{S}\setminus\{!\}$ of the binary relation defined on $\mathbb{S}$ by:

$$[s,s'] = \left\{ \ell \in L \;\middle|\; \exists (q \xrightarrow{\ell} q') \in \Delta_\tau \quad s.t. \quad q \in s \;\wedge\; q' \in s' \right\}$$

Note that $[s,!] = [!,\, s] = !$ for all $s \in \mathbb{S}$.

(c) Let $\oplus, \ominus : \mathbb{S} \times \mathbb{E} \to \mathbb{S}$ be the two operations defined as:

$$\begin{aligned} s \oplus e &= \left\{ q' \in \{0,1\} \;\middle|\; \exists (q \xrightarrow{\ell} q') \in \Delta_\tau \quad s.t. \quad q \in s \;\wedge\; l \in e \right\} \\ s \ominus e &= \left\{ q' \in \{0,1\} \;\middle|\; \exists (q' \xrightarrow{\ell} q) \in \Delta_\tau \quad s.t. \quad q \in s \;\wedge\; l \in e \right\} \end{aligned}$$

We recall that the signature of a rough set X is the map $X : E \to \mathbb{E}$ defined as

$$X(e) \;=\; \bigvee \left\{ [X(s), X(s')] \;\middle|\; s, s' \in S \quad s.t. \;\; s \xrightarrow{e} s' \in \Delta \right\}$$

Show that the rough regions of a transition system (S, E, Δ) coincide with the rough sets $X : S \to \mathbb{S}$ such that

$$X(s) \;=\; \bigvee \left\{ X(s') \oplus X(e) \;\middle|\; s' \xrightarrow{e} s \right\} \vee \bigvee \left\{ X(s') \ominus X(e) \;\middle|\; s \xrightarrow{e} s' \right\}$$

(d) Show that the rough region $\sigma(X)$ induced by a rough set X is the least refinement Y of X such that the following relations hold for every transition $s \xrightarrow{e} s'$ in Δ:

$$\begin{aligned} [Y(s), Y(s')] &\le Y(e) \\ Y(s) \oplus Y(e) &\le Y(s') \\ Y(s') \ominus Y(e) &\le Y(s) \end{aligned}$$

2.5. Relying on the algebraic results established in Prob. 2.4, we propose an incremental algorithm for computing the rough region $\sigma(X)$ induced by a given rough set X.

Let Y be a vector with two parts $Y : S \to \mathbb{S}$ representing a rough set and $Y : E \to \mathbb{E}$ representing the signature of a rough set. Initially $Y(s) = X(s)$ for all $s \in S$ and $Y(e) = ?$ for all $e \in E$. The two parts of the vector Y are updated by two functions which are applied in alternation until the algorithm terminates. These functions `update_sig`, resp. `update_conf`, update $Y : E \to \mathbb{E}$, resp. $Y : S \to \mathbb{S}$, and they use auxiliary data ∂S, resp. ∂E, indicating which states or events have received “fresh” values. Initially, $\partial S = X_\bullet \cup X_\circ$ and $\partial E = \emptyset$. These two functions may be described as follows.

1. `update_sig` updates $Y : E \to \mathbb{E}$ by recomputing, according to the formulas given in Prob. 2.4(e), the values $Y(e)$ for events e labelling transitions $s \xrightarrow{e} s'$ with $s \in \partial S$ or $s' \in \partial S$. If $Y(e)$ gets the undefined value $!$, then an exception is raised, the algorithm is stopped, and the incoherent rough region $Y(s) = !$ and $Y(e) = !$ for all $s \in S$ and $e \in E$ is returned as a result. Otherwise, `update_sig` returns as a result the set ∂E of events e whose values $Y(e)$ have actually been modified.

2. `update_conf` updates $Y : S \to \mathbb{S}$ by recomputing, according to the formulas given in Prob. 2.4(e), the values $Y(s)$ for states s which are sources or targets of transitions $s \xrightarrow{e} s'$ or $s' \xrightarrow{e} s$ labelled with $e \in \partial E$. If $Y(s)$ gets the undefined value $!$, then an exception is raised, the algorithm is stopped, and the incoherent rough region $Y(s) = !$ and $Y(e) = !$ for all $s \in S$ and $e \in E$ is returned as a result. Otherwise, `update_conf` returns as a result the set ∂S of states s whose values $Y(s)$ have actually been modified.

The algorithm terminates either when an exception is caught, or when one of the two update functions returns an empty set ∂E or ∂S. In the latter case the result returned is the vector Y, which represents the rough region $\sigma(X)$ and its signature.

Describe explicitly the sets ∂S and ∂E computed by the respective functions `update_sig` and `update_conf`, based on the results obtained in Prob. 2.4. Write a complete program that computes $\sigma(X)$ from X.

2.6. This exercise shows that, for deciding whether a coherent rough set X is terminal, it suffices to check the crossing properties of the events e such that $X(e) = -$ or $X(e) = +$ w.r.t. $X_\bullet$.

Given a coherent rough region X, show that $X_\bullet$ is a region if and only if

1. For every event $e \in E$ such that $X(e) = +$ (or $e \in X^+ \setminus ({}^\circ X \cup X^\perp)$), one of the following two situations occurs uniformly for all e-labelled transitions:

$$\begin{array}{lll} s \xrightarrow{e} s' & \Rightarrow & (s \in X_? \ \wedge \ s' \in X_\bullet) \\ s \xrightarrow{e} s' & \Rightarrow & (s \in X_\circ \ \wedge \ s' \in X_?) \vee (s \in X_? \ \wedge \ s' \in X_?) \end{array}$$

2. For every event $e \in E$ such that $X(e) = -$ (or $e \in X^- \setminus (X^\circ \cup X^\perp)$), one of the following two situations occurs uniformly for all e-labelled transitions:

$$\begin{array}{lll} s \xrightarrow{e} s' & \Rightarrow & (s \in X_\bullet \ \wedge \ s' \in X_?) \\ s \xrightarrow{e} s' & \Rightarrow & (s \in X_? \ \wedge \ s' \in X_\circ) \vee (s \in X_? \ \wedge \ s' \in X_?) \end{array}$$

2.7. In Section 2.3.2, it was explained how computing, for a rough region X, an over-approximation of the set of regions which are minimal in $R(X)$, i.e., a subset of regions $R'(X)$ such that $\min R(X) \subseteq R'(X) \subseteq R(X)$. The algorithm for computing $R'(X)$ may be summarized by the following recursive definition:

1. If X is incoherent, then $R'(X) = \emptyset$.
2. If X is a terminal rough region, then $R'(X) = \{X_\bullet\}$.
3. If X is not a terminal rough region,
   then choose some event e such that $X(e) \in \{-, +\}$ an let:
   a) $R'(X) = R'(X[e = +1]) \cup R'(X[e = 0])$ if $X(e) = +$
   b) $R'(X) = R'(X[e = -1]) \cup R'(X[e = 0])$ if $X(e) = -$

The purpose of this exercise is to design another algorithm which takes as input a rough region X and produces as an output the set $\min R(X)$ of the regions which are minimal in $R(X)$. We want to compute the set $\min R(X)$ incrementally, i.e., to decide *immediately* for each region produced whether it is minimal in $R(X)$ or not. The first step to take towards this goal is to reorder the sibling nodes of the binary tree $T(X)$ for ensuring that a region found at a leaf in a depth-first left-to-right traversal of this tree *cannot be larger than any region found subsequently*. Under this assumption, for computing the set $\min R(X)$, it suffices to check for each leaf of $T(X)$ that the region produced by this leaf is not larger than any region already produced (by leaves on its left). One may moreover improve on this naive technique by maintaining along the computation auxiliary informations needed to avoid useless comparisons. In order to decide which successor of a non terminal rough region Y should be visited first (i.e., put on the left), we need providing rough regions Y with extended signatures that classify the events of a transition system in a much finer way w.r.t. $Y_\bullet$ and $Y_\circ$. We define *extended signatures* as maps $Y : E \to \mathbb{E}'$ where $\mathbb{E}'$ is the lattice displayed in Fig. 2.14. Extended signatures are ordered

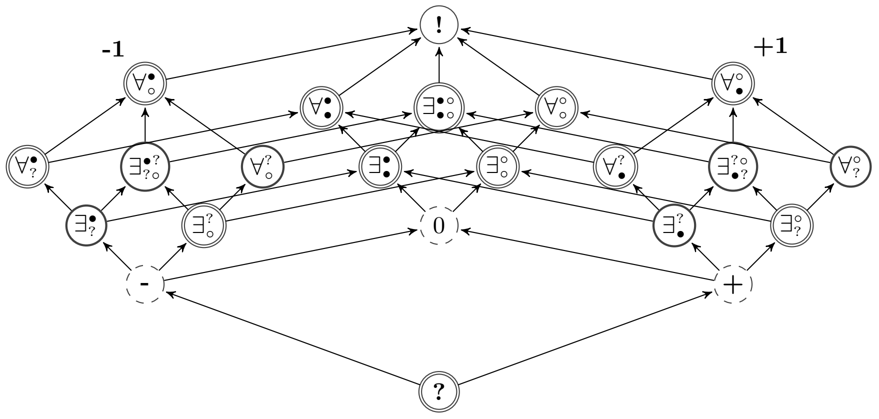


Fig. 2.14. a lattice $\mathbb{E}'$ for a refined classification of the events of a transition system

pointwise: $Y \leq Y' \Leftrightarrow \forall e \in E \quad Y(e) \leq Y'(e)$. The extended signature of a rough set $X = \langle X_\bullet, X_\circ \rangle$ is the least extended signature $X : E \to \mathbb{E}'$ such that the following relations hold for all $\alpha, \beta \in \{\circ, \bullet, ?\}$

$$\begin{array}{lll} \big(\exists s \xrightarrow{e} s' \;\cdot\; s \in X_\alpha \;\wedge\; s' \in X_\beta\big) & \Rightarrow & X(e) \geq \exists^\alpha_\beta \\ \big(\forall s \xrightarrow{e} s' \;\cdot\; s \in X_\alpha \;\wedge\; s' \in X_\beta\big) & \Rightarrow & X(e) \geq \forall^\alpha_\beta \end{array}$$

The rationale for considering $\exists^\alpha_\beta \leq \forall^\alpha_\beta$ may be found in the fact that uniform crossing relations w.r.t. $\{X_\bullet, X_?, X_\circ\}$ for all e-labelled transitions are more significant than similar crossing relations for just one e-labelled transition, owing to the assumption that every event e labels at least one transition. The elements $-$, 0, and $+$, represented inside dashed circles in Fig.2.14 do not belong to the image of any extended signature, and they have been added in $\mathbb{E}'$ just for convenience, see item (b) below.

(a) Show that if X is the extended signature of a rough region, then the following relations hold:

$$\begin{array}{lcl} X(e) \geq \exists^{\alpha}_{\beta} & \Leftrightarrow & \left(\exists s \xrightarrow{e} s' \quad\cdot\quad s \in X_\alpha \;\wedge\; s' \in X_\beta\right) \\ X(e) \geq \forall^{\alpha}_{\beta} & \Leftrightarrow & \left(\forall s \xrightarrow{e} s' \quad\cdot\quad s \in X_\alpha \;\wedge\; s' \in X_\beta\right) \end{array}$$

(b) Show that the signature of a rough region may be derived from its extended signature using the formulas

$$\begin{array}{ll} X^\circ & = \{e \in E \mid X(e) \geq -1\} \\ {}^\circ X & = \{e \in E \mid X(e) \geq +1\} \\ X^- & = \{e \in E \mid X(e) \geq -\} \\ X^+ & = \{e \in E \mid X(e) \geq +\} \end{array}$$

where -1 and $+1$ are used as aliases for $\forall^{\bullet}_{\circ}$ and $\forall^{\circ}_{\bullet}$ respectively (as indicated in Fig. 2.14).
(c) For any rough region X, prove the following relations (whose right members should be interpreted as conjunctions (i) and (ii)):

$$\begin{array}{lll} X(e) = \exists^{\bullet}_{?} & \Leftrightarrow (i) & \exists s \xrightarrow{e} s' \cdot s \in X_\bullet \text{ and } s' \in X_?\\ & (ii) & \forall s \xrightarrow{e} s' \cdot s \in X_\bullet \cup X_? \text{ and } s' \in X_?\\ X(e) = \exists^{?}_{\circ} & \Leftrightarrow (i) & \exists s \xrightarrow{e} s' \cdot s \in X_? \text{ and } s' \in X_\circ\\ & (ii) & \forall s \xrightarrow{e} s' \cdot s \in X_? \text{ and } s' \in X_\circ \cup X_?\\ X(e) = \exists^{?}_{\bullet} & \Leftrightarrow (i) & \exists s \xrightarrow{e} s' \cdot s \in X_? \text{ and } s' \in X_\bullet\\ & (ii) & \forall s \xrightarrow{e} s' \cdot s \in X_? \text{ and } s' \in X_\bullet \cup X_?\\ X(e) = \exists^{\circ}_{?} & \Leftrightarrow (i) & \exists s \xrightarrow{e} s' \cdot s \in X_\circ \text{ and } s' \in X_?\\ & (ii) & \forall s \xrightarrow{e} s' \cdot s \in X_\circ \cup X_? \text{ and } s' \in X_? \end{array}$$

Prove that $X(e) = \exists^{?\circ}_{\bullet ?}$ if and only if the following relations hold:

(i) $\exists s \xrightarrow{e} s' \cdot s \in X_?$ and $s' \in X_\bullet$
(ii) $\exists s \xrightarrow{e} s' \cdot s \in X_\circ$ and $s' \in X_?$
(iii) $\forall s \xrightarrow{e} s' \cdot (s \in X_? \wedge s' \in X_\bullet) \vee (s \in X_\circ \wedge s' \in X_?) \vee (s \in X_? \wedge s' \in X_?)$

Adapt the above statement to the case $X(e) = \exists^{\bullet ?}_{? \circ}$.
(d) Show that a rough region X is terminal if and only if, for every event e, the value of $X(e)$ is one of the elements of $\mathbb{E}'$ indicated with a double circle (hint: in this case, what does it mean to replace ? with $\circ$ uniformly in the values $X(e)$ for all $e \in E$?).
(e) Show that if X is a *non terminal* rough region, then for any event e:

$$\begin{array}{lll} e \in X^- \setminus (X^\perp \cup X^\circ) & \text{if and only if} & X(e) \in \{\exists^{\bullet}_{?}, \exists^{\bullet ?}_{? \circ}, \forall^{?}_{\circ}\} \\ e \in X^+ \setminus (X^\perp \cup {}^\circ X) & \text{if and only if} & X(e) \in \{\exists^{?}_{\bullet}, \exists^{? \circ}_{\bullet ?}, \forall^{\circ}_{?}\} \end{array}$$

The corresponding elements of $\mathbb{E}'$ are displayed with thick circles in Fig. 2.14.
(f) Propose a method for computing incrementally from $X = \langle X_\bullet, X_\circ \rangle$, for any rough set X, the extended signature of the rough region $\sigma(X)$.
(g) Show that one can always order the two successors X_1 and X_2 of a (non terminal) rough region X so that no region in $R(X_1)$ can be larger than any region in $R(X_2)$. Hint: use the following case analysis:

1. If $X(e) = \exists^{\bullet ?}_{? \circ}$, let $X_1 = \sigma(X[e := -1])$ and $X_2 = \sigma(X[e := 0])$, then two regions $r_1 \in R(X_1)$ and $r_2 \in R(X_2)$ cannot be compared, i.e., $r_1 \not\subseteq r_2$ and $r_2 \not\subseteq r_1$. A similar situation is found when $X(e) = \exists^{? \circ}_{\bullet ?}$.
2. If $X(e) = \exists^{\bullet}_{?}$, let $X_1 = \sigma(X[e := -1])$ and $X_2 = \sigma(X[e := 0])$, then for any two regions $r_1 \in R(X_1)$ and $r_2 \in R(X_2)$, $r_1 \not\supseteq r_2$. A similar situation is found when $X(e) = \exists^{?}_{\bullet}$.
3. If $X(e) = \forall^{?}_{\circ}$, then let $X_1 = \sigma(X[e := 0])$ and $X_2 = \sigma(X[e := -1])$, then for any two regions $r_1 \in R(X_1)$ and $r_2 \in R(X_2)$, $r_1 \not\supseteq r_2$. A similar situation is found when $X(e) = \forall^{\circ}_{?}$.

(h) If the sibling nodes of $T(X)$ are ordered as indicated in (g), and if one visits this tree by depth first and from left to right, checking that a terminal rough region $Y_\bullet$ belongs to $\min R(X)$ reduces to checking that this region is included in none of the regions produced so far. Justify the definition of the function $Rmin(X)$ given in Table 2.2 for computing incrementally the set $\min R(X)$. The first parameter of the function *visit* represents the currently visited node Y (rough region). The second and third parameters $\tau \subseteq E \times \wp(R(A))$ and $R \subseteq R(A)$ serve to reduce to a strict minimum the number of comparisons done for checking that a newly discovered region is minimal in $R(X)$, using the knowledge provided by the case analysis done in (g).

2.8. In order to compute the set $Rmin(A)$ of all minimal regions of an initialized transition system A, inducing an optimal net approximation of $SN_{Rmin(A)}(A)$ (see Sec. 2.1.1), we propose to mix the algorithm sketched after Rem. 2.50 with the algorithm presented in Table 2.2.

Recall that we should visit in sequence a family of trees $T_k = T(X_k)$ for $k = 1$ to $2n$, where $E = \{e_1, \ldots, e_n\}$ and for each $i \leq n$, X_i and X_{n+i} are the rough regions $[e_i = -1]$ and $[e_i = +1]$, respectively. In view of Prop. 2.49, if one applies the algorithm elaborated in Problem 2.7 to a fixed tree T_k, then we get as a result the set $R(X_k) \cap Rmin(A)$. So, we could content ourselves with using this algorithm to compute independently all sets $R(X_k) \cap Rmin(A)$ and then merge the results, but we want to do better.

The idea of the algorithm sketched after Rem. 2.50 is to avoid computing the tree $T(Y)$ for any node Y of a tree T_k such that $Y(e_j) = -1$ for some $j < k$ or $Y(e_j) = +1$ for some $j < k - n$, since in this case $R(Y) \subseteq \cup_{h<k} R(X_h)$. Thus, when visiting $T_1, \ldots, T_{k-1}$ in sequence, and exploring each of these trees by depth first and from left to right, one has necessarily met a set of nodes $Y_1, \ldots, Y_m$, possibly from different trees, such that $R(Y_j) \subseteq R(Y)$ for all $j \in \{1, \ldots, m\}$ and $R(Y) = \cup_{1 \leq j \leq m} R(Y_j)$. In particular, these relations

```
$Rmin(X) = visit(\sigma(X), \emptyset, \emptyset)$
$visit(Y, \tau, R) =$
begin
    if $Y$ a terminal configuration then
        if $r \subseteq Y_\bullet$ for some $r$ such that
          $r \in R$ or $\exists (e, R') \in \tau \cdot (Y(e) = \forall^\bullet_\bullet \wedge r \in R')$
        then return($\emptyset$) else return($\{Y_\bullet\}$)
    else choose some $e \in (Y^- \setminus (Y^\perp \cup Y^\circ)) \cup (Y^+ \setminus (Y^\perp \cup {}^\circ Y))$
        case $Y(e)$ of
            $\exists^{\bullet ?}_{? \circ}$ do $R_1 \leftarrow visit(\sigma(Y[e := -1]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(Y[e := 0]), \tau, R)$
                    return($R_1 \cup R_2$)
            $\exists^{? \circ}_{\bullet ?}$ do $R_1 \leftarrow visit(\sigma(Y[e := +1]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(Y[e := 0]), \tau, R)$
                    return($R_1 \cup R_2$)
            $\exists^\bullet_?$ do $R_1 \leftarrow visit(\sigma(Y[e := -1]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(Y[e := 0]), \tau \cup (e, R_1), R)$
                    return($R_1 \cup R_2$)
            $\exists^?_\bullet$ do $R_1 \leftarrow visit(\sigma(Y[e := +1]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(Y[e := 0]), \tau \cup (e, R_1), R)$
                    return($R_1 \cup R_2$)
            $\forall^?_\circ$ do $R_1 \leftarrow visit(\sigma(Y[e := 0]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(Y[e := -1]), \tau, R \cup R_1)$
                    return($R_1 \cup R_2$)
            $\forall^\circ_?$ do $R_1 \leftarrow visit(\sigma(Y[e := 0]), \tau, R)$
                    $R_2 \leftarrow visit(\sigma(X[e := +1]), \tau, R \cup R_1)$
                    return($R_1 \cup R_2$)
end
```

Table 2.2. pseudocode of the function Rmin

are satisfied for the set $trail(Y) = \{Y_1, \ldots, Y_m\}$ of already visited nodes Y_l with least signatures larger than the signature of Y. Whenever a node Y of a tree T_k such that $Y(e_j) = -1$ for some $j < k$ or $Y(e_j) = +1$ for some $j < k - n$, the idea is replace the effective visit to $T(Y)$ by a fictitious visit returning the same result, namely the sum of the sets of minimal regions already computed from all nodes $Y_l \in trail(Y)$. For this purpose, one must store at every node Y_l the set of minimal regions which have been computed when visiting this node. The result of the fictitious visit to the node X does not contribute to the final result, since it does not contain any new minimal region, but it may serve to avoid useless comparisons between new regions and already computed minimal regions, as was done in the algorithm elaborated in Problem 2.7. However, in that algorithm, the set of regions returned by function *visit* served two purposes: on the one hand, it contributed directly to the final result, and on the other hand it was transmitted as an argument for the visit to the sibling node. Here the two roles must be distinguished.

Write a modified version of the algorithm given in Table 2.2, incorporating the suggested adaptions.

2.9. Using the algorithm constructed in Prob. 2.8,
(a) compute the set of minimal regions of the transition system shown in Fig. 2.3 (on page 71),
(b) compute the set of minimal regions of the transition system shown in Fig. 2.4 (on page 72).

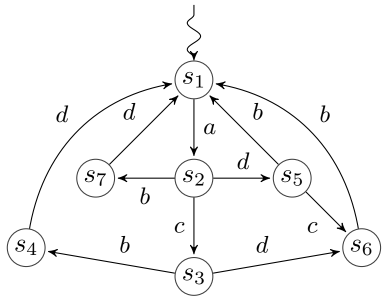
2.10. Construct an algorithm which, given a separation goal as input, produces an admissible set of minimal regions for this goal. For this purpose, use the results of Prob. 2.8.

2.11. Using the algorithm constructed in Prob. 2.10,
(a) compute an elementary net system with a reachability graph isomorphic to the transition system shown in Fig. 2.3 (on page 71),
(b) compute an elementary net system with a reachability graph isomorphic to the transition system shown in Fig. 2.4 (on page 72).

2.12. An improvement over the function $solve(\gamma)$ (defined on page 101) may be brought by avoiding to make an actual procedure call $visit(X, k, \gamma)$ at all times when it can be checked from $X_\bullet$, $X_\circ$ and the induced signature of X, that no region in $R(X)$ can solve any separation problem in γ.

We declare that a rough region X is *impractical* for a goal γ if and only if the following relations hold for all pairs $\{s, s'\}$ or $\{s, e\}$ or $\{e, e'\}$ of states s, s' and events e, e' (of A):
$(s, s') \in \gamma \wedge \{X(s), X(s')\} \subseteq \{0, 1\} \Rightarrow X(s) = X(s')$
$(s, e) \in \gamma \wedge X(e) = - \Rightarrow X(s) = 1$
$(e, e') \in \gamma \wedge \{X(e), X(e')\} \subseteq \{-1, 0, +1\} \Rightarrow X(e) = X(e')$
(a) Show that if X is impractical for γ, then $\gamma^R = \emptyset$ for $R = R(X)$.
(b) Show that the converse does not hold: $\gamma^R = \emptyset$ for $R = R(X)$ does not entail that X is impractical for γ.

2.13 (From [17]).
(a) Show that no net system can be language equivalent to the transition system displayed next.
(b) Split the event b so that the modified transition system may be realized by a net system. Draw the reachability graph of the net system.



3

Applications of Elementary Net Synthesis

In this chapter, we present applications of regions and elementary (or quasi-elementary) net synthesis in two different contexts. The first context is the design of speed independent circuits, a wide and complex field that encompasses many other topics than Petri nets. We refer the reader to the book [14] for a complete presentation of this field. We will limit ourselves to assess in Section 3.1 the roles played by regions and net synthesis for affording solutions to the Complete State Coding problem (CSC). This crucial problem motivated indeed the inclusion of a net synthesis procedure in PETRIFY, a general design tool for asynchronous circuits presented in [14]. However, different methods based on net unfoldings and SAT solvers are now preferred to net synthesis for solving the CSC problem. The second context of application of net synthesis that we examine is process mining, and more precisey, business process mining. This field encompasses many aspects little related or not related at all with Petri nets. We refer the reader to the book [35] for a complete presentation of process mining, a topic which is again too large to be surveyed here. We will limit ourselves to give in Section 3.2 a comparative assessment of the α algorithm, embedded in the tool PROM, and of other workflow net synthesis algorithms based on regions. Process mining algorithms based on the synthesis of P/T-nets are left for further consideration in part II of this book.

3.1 Regions in the Design of Speed Independent Circuits

Regions and the region-based synthesis of Elementary Nets play an important role in PETRIFY, a general design tool for Speed Independent Circuits presented in a series of papers and in the book [14]. In the design of an asynchronous circuit, the intended behaviour may be represented by an initialized transition system called a *State Graph* (SG), whose transitions are labelled with events $x+$ or $x-$ to indicate that a signal x is raising or falling. There are *input signals* switched by the environment, and *internal signals* and *output signals* switched by the circuit (internal signals and output signals are also called *non-input signals*). Schematically, there are two phases of design, both often requiring the insertion of internal signals which makes an automated tool like PETRIFY quite necessary. Given an arbitrary SG, the first phase consists of refining this description by inserting new signals until one meets conditions that guarantee the existence of a speed independent circuit behaving as described. Given a library of gates, the second phase consists of implementing the boolean next-state functions of the internal and output signals as combinations of the functions of these gates without making assumptions on switching delays. During this functional and combinational decomposition, new signals appear naturally when putting wires between gates. Regions and region-based synthesis are used in the first phase of the design, which is considered below. Our account relies mostly on [12], [13] and [14].

Definition 3.1. *Let $X = X_I \cup X_O$ be a finite set of* binary signals*, where X_I is the set of input signals and X_O is the set of internal or output signals. A* State Graph $(A, X, \lambda_S, \lambda_E)$ *is an initialized transition system $A = (S, E, T, s_0)$ equipped with two labelling maps $\lambda_S : S \to \{0,1\}^X$ (called the* state assignment *function) and $\lambda_E : E \to X \times \{+,-\}$ (called the* event assignment *function). The* flat version *of the state graph A is the initialized transition system $A' = (S, X \times \{+,-\}, T', s_0)$ defined by $T' = \left\{ s \xrightarrow{\lambda_E(e)} s' \;\middle|\; s \xrightarrow{e} s' \in T \right\}$. We recall that A' is* deterministic *if, for any $x* \in X \times \{+,-\}$ and for any state $s \in S$, $s \xrightarrow{x*} s_1$ and $s \xrightarrow{x*} s_2$ entail $s_1 = s_2$. A' is* commutative *if, for any $x*, y* \in X \times \{+,-\}$ and for any state $s \in S$, $s \xrightarrow{x*} s_1 \xrightarrow{y*} s_2$ and $s \xrightarrow{y*} s_3 \xrightarrow{x*} s_4$ entail $s_2 = s_4$. A state graph is deterministic, resp.commutative, if its flat version is deterministic, resp. commutative. Two state graphs are said to be* equivalent *if their flat versions are isomorphic.* $\diamondsuit$

Given a state graph SG, assume that it describes actually the behaviour of a speed independent circuit, that is to say, a set of gates and wires whose functional behaviour does not depend upon delays of gates. Then each sequence of transitions $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} s_{n+1}$ in SG describes a possible evolution of the circuit, where every signal x has the initial value $\lambda_S(s_0)(x)$. A transition $s \xrightarrow{e} s'$ labelled with $\lambda_E(e) = x+$ (resp. $x-$) represents a raising (resp. falling) edge of the signal x. In order that labels of states indicate values of signals compatible with the induced switching sequences, the following condition must be satisfied:

Consistency: for every transition $s \xrightarrow{e} s'$:

1. if $\lambda_E(e) = x+$ then $\lambda_S(s)(x) = 0$ and $\lambda_S(s')(x) = 1$,
2. if $\lambda_E(e) = x-$ then $\lambda_S(s)(x) = 1$ and $\lambda_S(s')(x) = 0$,
3. otherwise $\lambda_S(s)(x) = \lambda_S(s')(x)$.

Two other conditions must be satisfied. In order that the described behaviour can be implemented by gates, it is necessary that the enabling of every non-input signal x to switch (from 0 to 1 or from 1 to 0) in a state s depends functionally upon the vector $\lambda_S(s) \in \{0,1\}^X$, since otherwise no next state function could be defined for the signal x. This condition is called CSC (Complete State Coding condition). In order that the described behaviour can be obtained from a set of gates and wires without making assumptions on the delays of gates, it is necessary that the switching of a non-input signal cannot disable or be disabled by the switching of another non-input signal. This condition is called Output Persistency. These two conditions upon the flat version of the state graph may be expressed formally as follows:

Output Persistency: for all $x*, y* \in X \times \{+,-\}$ with $x* \neq y*$, for all transitions $s \xrightarrow{x*} s'$ and $s \xrightarrow{y*} s"$, if x or is a non-input signal, or y is a non-input signal, then there exists some transition $s" \xrightarrow{x*}$.

Complete State Coding: for every non-input signal $x \in X_O$, and for all states $s', s" \in S$ with $\lambda_S(s') = \lambda_S(s")$, if $s' \xrightarrow{x*}$ with $x* = x+$ or $x* = x-$, then there exists some transition $s" \xrightarrow{x*}$.

The state graph generated by a speed-independent circuit is finite, consistent, output persistent and satisfies CSC. Conversely, these four conditions guarantee that a state graph can be *implemented* by a speed independent circuit. If a state graph SG, given as the specification of the behaviour of a speed independent circuit, is not consistent, or if it is not output persistent, then this must be fixed by a redesign. In contrast, the CSC property can sometimes be enforced on a state graph by inserting auxiliary signals while preserving consistency and output persistency. Let SG and SG' be two state graphs, where SG' has been obtained from SG by signal insertions (a precise definition will be given later). A speed independent circuit that implements SG' is considered to be also an implementation of SG if the flat versions of the two state graphs are language equivalent (when projecting the sequences of labels generated by SG' on $X \times \{+,-\}$ where X is the set of signals of SG), and for every sequence leading SG' to a sink state (i.e., a state which is not the source of any transition), the projection of this sequence on $X \times \{+,-\}$ leads SG to a sink state. It has been shown in [15] that signal insertions are always sufficient to enforce the CSC condition on (flat versions of) state graphs derived from Petri nets (more precisely, from Signal Transition Graphs which are labelled Petri nets) provided these state graphs are consistent, output persistent, deterministic and commutative.

Definition 3.2. *A* State Transition Graph (N, X, λ) *over the set of signals X is a Petri net system $N = (P, T, F, M_0)$ equipped with a labelling map $\lambda : T \to X \times \{+,-\}$.* ♢

Of particular interest are the STG whose reachability graphs can be transformed to a consistent SG in a unique way by labelling the reachable markings in $X \to \{0,1\}$. This uniqueness property may be enforced by supplying in the net, for each signal $x \in X$, two auxiliary places px and px' with the flow relations $F(x+, px), F(px', x+)$ and $F(x-, px'), F(px, x-)$. Given an STG (N, X, λ) with reachability graph $RG(N)$, deciding whether exists some state coding map $\lambda_S : RS(N) \to \{0,1\}^X$ that transforms $RG(N)$ into a state graph satisfying CSC is much more difficult. This problem is indeed co-NP-complete in the size of the STG [23] (co-NP-completeness means that it is NP-complete to check the non-existence of a state coding map ensuring CSC). Moreover, the complexity of the problem is not decreased if one restricts STG to (1-safe) live or acyclic marked graphs (a marked graph is a net in which every place has at most one input transition and at most one output transition).

In order to eliminate CSC conflicts, i.e., pairs of markings M and M' with identical binary codes ($\lambda_S(M) = \lambda_S(M')$) but different sets of enabled output signals $x+$ or $x-$, a direct *transformation* of STG by signal insertions has been defined in [13] and [14]. The transformation consists of adding for each place p of the STG a new signal πp, for each transition t of the STG a new signal τt, and to proceed as follows. For every transition t with input places $p_{i1}, \ldots, p_{in}$ and ouptut places $p_{o1}, \ldots, p_{oj}$, insert between t and its output places a subnet with fresh places and fresh transitions labelled as indicated in Fig. 3.1 (thick edges represent places like in marked graphs, e.g, the edge between $\pi p_{oj}+$ and $\pi p_{in}-$ represents an empty place p with the flow relations $F(\pi p_{oj}+, p)$ and $F(p, \pi p_{in}-)$). Note that several transitions may bear the same label $\pi p+$ or $\pi p-$ in the the resulting STG.

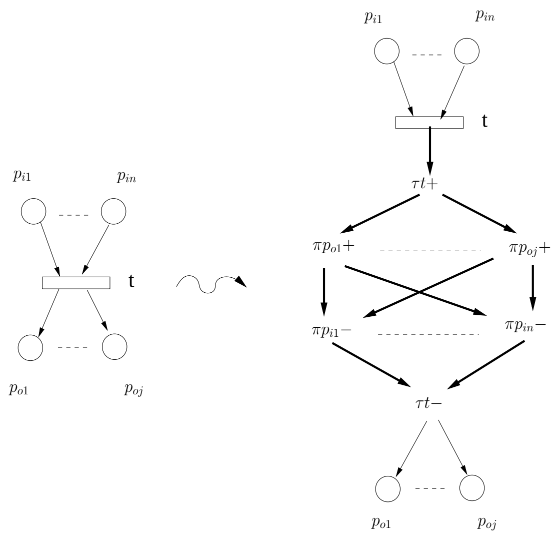


Fig. 3.1. Transformation for a transition t

The transformation sketched in Fig. 3.1 enforces the CSC property on state graphs of STG and it preserves consistency, output persistency, determinism and commutativity. Moreover, it produces an STG with an unchanged language (when the generated words are projected on the original alphabet $X \times \{+,-\}$), and it does not introduce new sink states. Nervertheless, this transformation has mostly theoretical interest according to the authors of [13] and [14], because it results in STG with lots of internal signals that may lead to impractical and inefficient circuits. A similar remark applies to the slightly different transformation presented in [11], which works linearly in the size of the STG. For eliminating CSC conflicts, these authors have therefore elaborated and implemented in PETRIFY other methods that work directly on (flat versions of) state graphs SG without synthesizing first an STG from the SG. These alternative methods, which have been proved to be complete for state graphs of STG but not for arbitrary SG, proceed by repeated insertions of new signals induced by bipartitions of the state space of the SG. Before we proceed to describe precisely the signal insertions in SG, we would like to make a first assessment.

At this stage, one can identify two different roles of net synthesis in PETRIFY *and the design of speed independent circuits. First, given a consistent and output persistent SG, net synthesis may serve to check whether the given SG is equivalent to the state graph of some STG, i.e., whether it belongs to the domain in which signal insertions are a complete method for eliminating CSC conflicts. Second, when a consistent and output persistent SG satisfying CSC has been obtained by signal insertions, net synthesis may help to produce an equivalent but more compact representation in the form of an STG.*

We describe now the procedure for inserting a signal after bipartioning a State Graph. In a preliminary stage, we describe two alternative procedures for inserting an event by a non-empty subset of states in an initialized transition system $A = (S, E, T, s_0)$. The following definitions are illustrated in Fig. 3.2.

Definition 3.3. *Inserting* $e' \notin E$ after $Z \subseteq S$ *means refining* A *to* $A' = (S \cup Z', E \cup \{e'\}, T', s_0)$ *where* Z' *is a set of fresh states in bijection with* Z *(let the bijection send* $s \in Z$ *to* $s' \in Z'$*) and* T' *is the set of transitions as follows. First,* T' *contains a fresh transition* $s \xrightarrow{e'} s'$ *for each state* $s \in Z$*. Second,* T' *contains all transitions* $s_1 \xrightarrow{e} s_2$ *in* T *with* $s_1 \notin Z$ *or* $s_2 \in Z$*. Third, for each transition* $s_1 \xrightarrow{e} s_2$ *in* T *with* $s_1 \in Z$*,* T' *contains the transition* $s_1' \xrightarrow{e} s_2$ *if* $s_2 \notin Z$*, or the transition* $s_1' \xrightarrow{e} s_2'$ *if* $s_2 \in Z$*.* ♢

Definition 3.4. *Inserting* $e' \notin E$ before $Z \subseteq S$ *means refining* A *to* $A' = (S \cup Z', E \cup \{e'\}, T', s_0)$ *where* Z' *is a set of fresh states in bijection with* Z *(let the bijection send* $s \in Z$ *to* $s' \in Z'$*) and* T' *is the set of transitions as follows. First,* T' *contains a fresh transition* $s' \xrightarrow{e'} s$ *for each state* $s \in Z$*. Second,* T'

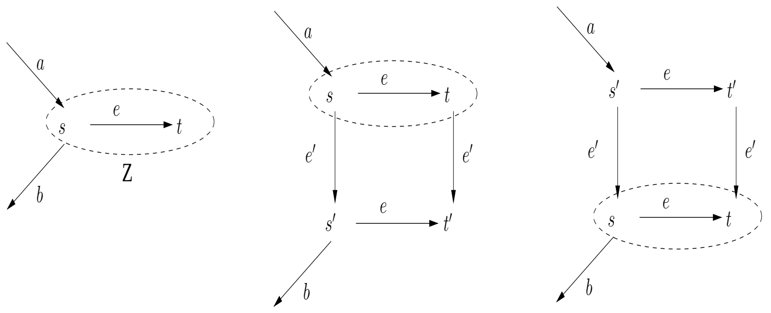


Fig. 3.2. Inserting e' after Z or before Z

contains all transitions $s_1 \xrightarrow{e} s_2$ in T with $s_2 \notin Z$ or $s_1 \in Z$. Third, for each transition $s_1 \xrightarrow{e} s_2$ in T with $s_2 \in Z$, T' contains the transition $s_1 \xrightarrow{e} s'_2$ if $s_1 \notin Z$, or the transition $s'_1 \xrightarrow{e} s'_2$ if $s_1 \in Z$. $\diamondsuit$

Now consider a state graph $SG = (A, X, \lambda_S, \lambda_E)$ where $A = (S, E, T, s_0)$. For any subset of states $B \subseteq S$, define the *exit border* of B as the set $EB(B) = \{s \in B \mid \exists e \in E\ \exists s' \in S : s \xrightarrow{e} s' \wedge s' \notin B\}$ and the *input border* of B as the set $IB(B) = \{s \in B \mid \exists e \in E\ \exists s' \in S : s' \xrightarrow{e} s \wedge s' \notin B\}$. *Inserting a signal x by the exit borders of a partition $\{B, S \setminus B\}$* means inserting simultaneously in A an event $e-$ after the exit border $EB(B)$ and an event $e+$ after the exit border $EB(S \setminus B)$, with $\lambda_E(e-) = x-$ and $\lambda_E(e+) = x+$ or vice-versa. *Inserting a signal x by the input borders of a partition $\{B, S \setminus B\}$* means inserting simultaneously in A an event $e+$ before the input border $IB(B)$ and an event $e-$ before the input border $IB(S \setminus B)$, with $\lambda_E(e+) = x+$ and $\lambda_E(e-) = x-$ or vice-versa. In any case, if SG is a consistent state graph, then at most one extension of the map λ_S can preserve consistency. The conditions in which this occurs are stated in the following proposition, proved in [13] and [14].

Proposition 3.5. *Inserting a signal x by the exit (resp. input) borders of a partition $\{B, S \setminus B\}$ preserves consistency if and only the exit (resp. input) borders of B and $S \setminus B$ are well-formed in the following sense:*

1. *$EB(B)$ is well-formed if every transition $s \xrightarrow{e} s'$ from $s \in EB(B)$ reaches some state $s' \in (S \setminus B) \cup EB(B)$, and similarly for $EB(S \setminus B)$;*
2. *$IB(B)$ is well-formed if every transition $s' \xrightarrow{e} s$ to $s \in IB(B)$ originates from some state $s' \in (S \setminus B) \cup IB(B)$, and similarly for $IB(S \setminus B)$.* $\square$

Thus, whenever a CSC conflict occurs between two states s_1, s_2 of a state graph SG, if $s_1 \in B$ and $s_2 \in S \setminus B$ for some partition $\{B, S \setminus B\}$ satisfying the conditions of the proposition, this CSC conflict can be reduced by inserting a new signal x. However, such signal insertions do not necessarily preserve output-persistency and commutivity of state graphs. Relying on the completeness of the transformation sketched in Fig. 3.1, it may be shown that, given any STG with a deterministic, commutative, output persistent and consistent state graph SG, one can reduce all CSC conflicts by repeated signal insertions preserving these properties. In practice, it remains however to find at each stage some partition of the set of states such that a signal insertion at the borders of this partition reduces some violation of the CSC property without compromising the other properties. This leads us to make a second assessment about the role of regions in the design of speed independent circuits.

In PETRIFY, *regions play a third role by guiding the choice of the partitions used to reduce CSC conflicts by signal insertions at their borders.*

The following definition and proposition clarify the connections between regions and CSC-conflicts.

Definition 3.6. *Given an initialized transition system* (S, E, T, s_0)*, a region* $r \subseteq S$ *is* exit persistent *if, for all states* $s \in r$, $s \xrightarrow{e_1} s_1$ *and* $s \xrightarrow{e_2} s_2$ *with* $s_1 \notin r$ *and* $s_2 \in r$ *entail the existence of some transition* $s_2 \xrightarrow{e_1}$. ♢

Proposition 3.7. *Given a consistent, output persistent, deterministic and commutative state graph* $(A, X, \lambda_S, \lambda_E)$*, suppose that the event-state separation property* $ESSP$ *holds in the underlying transition system* $A = (S, E, T, s_0)$ *(in the terminology of [14], A is excitation closed). Let* $B \subseteq S$ *be a region of* A*. If* B *and the complementary region* $S \setminus B$ *are exit persistent and they have well-formed exit borders, then inserting a new signal by the exit borders of the partition* $(B, S \setminus B)$ *produces a state graph with the same properties.* □

As consistency, output persistency, determinism and commutativity are preserved under state graph equivalence, one can freely split the events of A until the property $ESSP$ holds.

Beside signal insertions directly based on regions (Prop. 3.7), PETRIFY uses also for signal insertions sophisticated heuristics using intersections of regions and event enabling sets of states (called excitation regions). It is not clear however that such heuristics are complete, whereas signal insertions operating the transformation of STG indicated in Fig. 3.1 are complete.

To conclude this section, we should indicate that the use of regions and net synthesis for solving CSC conflicts in STG has now mainly historical interest, since other methods using finite complete prefixes of STG (instead of reachability graphs) and SAT solvers (instead of regions and net synthesis) give better results in practice (see, e.g., [27] for more on this topic).

3.2 Elementary Net Synthesis and Process Mining

The purpose of *process mining* [35] is to construct or to reconstruct from an event log a business process model that can generate this event log. The game is to dig out of event logs sufficient informations on the structure of their generating model. As a technique for *model discovery*, process mining has some connections with machine learning.

Process mining may be used for the purpose of modelling. For instance, after collecting over a long period of time informations on the health history of many patients, including the diagnosis and treatment steps, one may want to extract from this record an accurate model of the workflow system of an hospital. *Reverse engineering*, which consists of reconstructing from representative use cases an existing but partially unknown system, is another activity of model discovery that can be achieved by process mining. According to [35], process mining can be used alternatively for *conformance* checking or *enhancement* of business process models. For instance, *process-aware systems* record run-time informations used to detect discrepancies between expected and actual behaviours and to refactor these systems.

In this section we focus our attention on model discovery. We fix a subclass of net systems, the so-called *workflow nets*, as the class of target models for process mining. We describe in Sect. 3.2.1 the model of workflow nets and the general problem of discovering workflow nets from event logs. We present and assess in Sect. 3.2.2 the α algorithm for mining workflows from event logs [38]. We consider in Sect. 3.2.3 an alternative workflow mining algorithm based on regions [9, 10].

3.2.1 Discovering Workflow Nets from Event Logs

A log is a finite set of independent execution sequences of a workflow system. Given a log of the system to be discovered (i.e., constructed or reconstructed), each event reported in this log refers to an *activity*, i.e., a particular step in the workflow system, and to a specific *case*, i.e., a process instance. Additional informations pertaining to events are generally included, for instance a time stamp or the identity of the performer. Since cases have little or no connection with one another, and time stamps serve only to indicate the correct ordering of the activities, an event log may be abstracted to a *set of sequences of activities*. Each sequence represents all activities of a case from the time when it enters the system till the time when it leaves the system.

As the target representation of process mining, we consider a subclass of (quasi-elementary) net systems, called *workflow nets*. The goal of process mining is to synthezise from an event log a workflow net that can reproduce all sequences of activities traced in this log, from the inception of a case to its termination.

Example 3.8. A workflow net is displayed in Fig. 3.3. This net specifies all

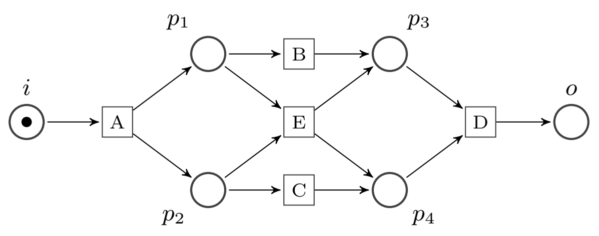


Fig. 3.3. a workflow net

possible behaviours of an isolated case in a workflow system. In other words, the firing sequences of the workflow net represent bijectively all activities pertaining to a case from the time when the case enters into the system (the input place i is marked) until the case terminates and exits from the system (the output place o is marked). The possible sequences of activities are thus $ABCD, ACBD, AED$. □

A workflow net contains an *input* place i and an *output* place o. The input place i is initially marked, and this initial marking represents the entry of a new case in the system. The output place o gets marked when the case comes to completion, and this final marking represents the exit of the case from the system. The current marking of the workflow net represents the current status of a case. It is assumed that the execution of a case can always reach termination, and that it cannot interfere with the execution of any subsequent case. The latter property, called *soundness* in [38], can be formalized as follows: when the output place is marked, all other places must be empty. The marking in which the output place and no other place is marked is called the *terminal marking*. Therefore, in a sound workflow net, the terminal marking must be reachable from any other reachable marking.

Moving the token from the output place back to the initial place is a way to simulate the termination of a case and the inception of a new case. A workflow net with such an implicit feedback may be seen as a cyclic generator, that can iterate in sequence all scenarios of execution of all cases. Instead of adding a feedback transition from the output place to the input place, one might as well coalesce the input place and the output place (confusing thus the initial and terminal markings). With this representation, the two crucial properties of workflow nets N may be reformulated equivalently as follows: *(i)* the net N' obtained by coalescing the input place and the output place of N is *reversible* (the initial marking may be reached from any reachable marking) and *(ii)* the initial (or terminal) marking is the only reachable marking of N' containing the input (or output) place. This is essentially the definition of workflow nets which we adopt below.

Before stating this definition, let us recall that a net system is live if, for any transition t and for any reachable marking M, the transition t is enabled in some marking reachable from M.

Definition 3.9. *A* workflow net *is a contact-free and connected net system $N = (P, T, F, M_0)$ where P contains an input place i and an output place o (the remaining places $p \in P \setminus \{i, o\}$ are called* inner places*), such that the following conditions hold:*

1. $^\bullet i = o^\bullet = \emptyset$
2. $(\forall p \in P \setminus \{i, o\}) \quad ^\bullet p \neq \emptyset \quad \wedge \quad p^\bullet \neq \emptyset$
3. $M_0 = \{i\}$.
4. *The* closed *net system $N' = (P', T, F', \{\iota\})$ obtained from N by replacing places i and o with a unique place ι such that $^\bullet\iota = {}^\bullet o$ and $\iota^\bullet = i^\bullet$ is live and reversible, and its initial marking $M'_0 = \{\iota\}$ is the unique reachable marking of N' in which the place ι is marked.* ♢

In view of Def. 3.9, workflow nets are initially live, in conformity with the general assumption that all net systems considered in this book are initially live (i.e., free from dead transitions). Def. 3.9 is an equivalent reformulation of the definition of *sound workflow nets* given in [38] (contact-free quasi-elementary net systems are equivalent to one-safe net systems).

Remark 3.10. Verifying the following properties is left as Exercise 3.1.

1. $\forall t \in T \quad ^\bullet t \neq \emptyset \ \wedge \ t^\bullet \neq \emptyset$.
2. The closed net system N' is strongly connected.
3. $i^\bullet = \{t \in T \mid \exists u \in T^* \quad t \cdot u \in L(N)\}$.
4. $^\bullet o = \{t \in T \mid \exists u \in T^* \quad u \cdot t \in L(N)\}$.
5. $(\forall p \in P \setminus \{i, o\}) \quad ^\bullet p \cap {}^\bullet o = \emptyset \quad \wedge \quad p^\bullet \cap i^\bullet = \emptyset$. □

The central notion of workflow logs is defined below.

Definition 3.11. *Given a workflow net $N = (P, T, F, M_0)$, the* full log *of N is the language $\mathcal{L}(N) = \{u \in T^* \mid \{i\} [u\rangle \{o\}\}$, i.e., the full log of N is the set of all firing sequences from the initial marking $\{i\}$ to the final marking $\{o\}$. A* log *of N is any subset $W \subseteq \mathcal{L}(N)$ such that every transition $t \in T$ occurs in at least one execution sequence in W. A* workflow log *is any log of a workflow net.* ♢

Since workflow nets have no dead transitions, the full log of a workflow net is actually a log of this workflow net. In view of Remark 3.10, it can easily be checked that any workflow log is a *w-language* according to the definition below.

Definition 3.12. *A language $W \subseteq T^*$ is a w-language if the following conditions hold:*

1. *Every transition $t \in T$ occurs in at least one word in W.*

2. $W \subseteq I_W(T \setminus I_W)^*$ *where* $I_W = \{t \in T \mid \exists u \in T^* \quad t \cdot u \in W\}$*, i.e., transitions occurring in the first position of some word in* W *cannot occur in different positions in any other word in* W*.*
3. $W \subseteq (T \setminus O_W)^* O_W$ *where* $O_W = \{t \in T \mid \exists u \in T^* \quad u \cdot t \in W\}$*, i.e., transitions occurring in the last position of some word in* W *cannot occur in different positions in any other word in* W*.* ♢

In view of conditions (2) and (3) in Def. 3.12, any w-language W is a prefix language, i.e., a word in W cannot be a prefix of a strictly longer word in W. In view of item (4) in Def. 3.9, every firing sequence of a workflow net can be extended to a firing sequence that ends in the final marking $\{o\}$ of this net. Therefore, once can state the following remark.

Remark 3.13. $\mathcal{L}(N) = max(L(N))$ and $L(N) = pref(\mathcal{L}(N))$ for any workflow net N where, for any $W \subseteq T^*$, $max(W) = \{u \in W \mid u \cdot v \in W \Rightarrow v = \varepsilon\}$ and $pref(W) = \{u \in T^* \mid \exists v \in T^* \, u \cdot v \in W\}$ for $W \subseteq T^*$. □

A fundamental assumption of process mining is that the set of execution sequences $W \subseteq T^*$ taken as input is actually a log of some unknown workflow net N, that the mining algorithm μ should try to reconstruct, i.e., $W \subseteq \mathcal{L}(N)$ and hopefully $N \cong \mu(W)$ where $\mu(W)$ is the result produced by the mining algorithm.

Definition 3.14. *A workflow net* N *is* discovered *from one of its log* $W \subseteq \mathcal{L}(N)$ *by a process mining algorithm* μ *if* $N \cong \mu(W)$*, meaning that the workflow nets* N *and* $\mu(W)$ *coincide up to a bijective renaming of places (they are isomorphic). A workflow net* N *is said to be* μ-reconstructible *if it can be discovered from its full log, i.e.,* $N \cong \mu(\mathcal{L}(N))$*.* ♢

Example 3.15 (Exple. 3.8 continued).
Consider the log $\{ABCD, ACBD, AED\} \subseteq \mathcal{L}(N)$ of the workflow net N shown in Fig. 3.3. Every execution sequence in this log starts with event A and ends with event D. In between, one is left the choice to perform either the event E or the events B and C, which are concurrent since they occur in the log in both orders BC and CB. Using these structural informations extracted from the log, the two process mining algorithms examined in the end of the section, the α algorithm and the region based mining algorithm, are able to discover the workflow net N from the considered log. All three execution sequences in this log are actually needed: every activity ought to be reported in at least one execution sequence(thus AED is needed), and for any pair of concurrent events, at least two traces exhibiting the two possible orderings are needed (thus $ABCD$ and $ACBD$ are needed). □

The set of all execution sequences of a workflow net may grow exponentially with the number of events, owing to their possible concurrency. Therefore the execution sequences reported in an event log usually form a small but hopefully representative set of samples of all possible behaviours. In order to discover a workflow net N from some small log $W \subset \mathcal{L}(N)$, a process mining algorithm μ must carry out some non-trivial generalization over the traces in this log. Henceforth, we shall assume that a process mining algorithm μ is always defined as the composition $\mu = Syn \circ Abs$ of an abstraction function Abs and a synthesis function Syn. The role of the function Abs is to extract from a log W the relevant relations between the events that occur in this log. The role of the function Syn is to reflect, as faithfully as possible, these relations in the structure of a synthesized net system. We assume that for any log W, $\mu(W)$ is a well-defined (quasi-elementary) net system, *but we do not assume that $\mu(W)$ is always a workflow net.* We do not either set injectivity as a requirement on the synthesis function Syn, because this would hamper our planned presentation of the α algorithm.

Definition 3.16. *Given a workflow net N and an abstraction function Abs defined on $\wp(\mathcal{L}(N))$, a w-language $W \subseteq \mathcal{L}(N)$ is a* complete *log of N w.r.t. Abs if $Abs(W) = Abs(\mathcal{L}(N))$. If $\mu = Syn \circ Abs$, W is then said to be a μ-complete log of N.* $\diamondsuit$

Remark 3.17. Assuming that $W \subseteq T^*$ is a μ-complete log of a(n unknown) workflow net N, the following conditions are equivalent:

1. N is μ-reconstructible: $N \cong \mu(\mathcal{L}(N))$;
2. N can be discovered from W: $N \cong \mu(W)$;

So, if $W \subseteq T^*$ is a μ-complete log of some μ-reconstructible workflow net N, then $N \cong \mu(W)$. On the other hand, $W \subseteq T^*$ is a μ-complete log of $\mu(W)$ if and only if:

1. $\mu(W)$ is a workflow net and $W \subseteq \mathcal{L}(\mu(W))$.
2. $Abs(W) = Abs(\mathcal{L}(\mu(W)))$.

and then $\mu(W)$ is μ-reconstructible. $\square$

In fact, a workflow net N is μ-reconstructible if and only if it may be discovered by μ from some complete log W (of N). Process discovery amounts to the following:

Problem: decide whether a given w-language $W \subset T^*$ is a μ-complete log of some μ-reconstructible workflow net.
Solution: compute the net system $N = \mu(W)$, check that *(i)* N is a workflow net, *(ii)* $W \subseteq \mathcal{L}(N)$, i.e., the synthesized net can reproduce every execution sequence in W, and *(iii)* $Abs(W) = Abs(\mathcal{L}(\mu(W))$. If this is the case, then W is a complete log of the workflow net $N = \mu(W)$, and $N \cong \mu(\mathcal{L}(N))$. Otherwise, no solution exists.

If the synthesis function Syn is non-injective, then the relation $\mu(W') = \mu(\mathcal{L}(N))$ may also hold for an incomplete log W' of N, and in particular for a log W' such that $W \subseteq W' \subseteq \mathcal{L}(N)$. Worse, it may occur that a workflow net N, discovered by μ from some complete log W, cannot be discovered from some incomplete log W' such that $W \subseteq W' \subseteq \mathcal{L}(N)$. In order to avoid such situations, we shall always assume in the sequel that abstraction functions *Abs* are convex according to the following definition.

Definition 3.18. *Given a workflow net N, an abstraction function Abs defined on $\wp(\mathcal{L}(N))$ is convex if every log of N that contains a complete log is also a complete log, i.e., $W \subseteq W' \subseteq \mathcal{L}(N)$ and $Abs(W) = Abs(\mathcal{L}(N))$ entail $Abs(W') = Abs(\mathcal{L}(N))$.* ♢

Other desirable properties of process mining algorithms are discussed below. If the abstraction function *Abs* is the identity on $\wp(\mathcal{L}(N))$, i.e., it does not entail any abstraction, then $\mathcal{L}(N)$ is the unique complete log of N. More generally, if the abstraction is too weak, the language of the net $\mu(W)$ may often be a tight over-approximation of W, much smaller than $\mathcal{L}(N)$ (the logs are supposed to report only a small fragment of the set of all possible behaviours). In that case, $\mu(W)$ is an *overfitted* model[36]. Overfitting occurs when the process mining algorithm does not carry out sufficient generalization over the set of traces in the log. On the contrary, good process mining algorithms should be sober according to the following definition.

Definition 3.19. *A process mining algorithm $\mu = Syn \circ Abs$ is* sober *if its abstraction function Abs is convex and the minimal size of complete logs of workflow nets is asymptotically negligible w.r.t. the size of their languages.* ♢

Sobriety means that one may generally assume that a workflow log is μ-complete as soon as it contains a reasonable number of execution sequences. Process reconstruction amounts to the following:

Problem given a sober mining algorithm μ and a reasonably large w-language $W \subset T^*$, decide whether W is a log of some μ-reconstructible workflow net.
Solution compute the net system $N = \mu(W)$, check that *(i)* N is a workflow net, and *(ii)* $W \subseteq \mathcal{L}(N)$, i.e., the synthesized net can reproduce each execution sequence in W. If this is the case, then W is a complete log of the workflow net $N = \mu(W)$, and $N \cong \mu(\mathcal{L}(N))$. Otherwise, no solution exists.

Sober mining algorithms avoid producing overfitted models (models that do not generalize enough). On the other hand, producing *underfitted* models allowing too many behaviours results in mining algorithms with low expressivity according to the following definition.

Definition 3.20. *The* expressivity *of a process mining algorithm μ is given by the class of the μ-reconstructible workflow nets.* ♢

To sum up, one needs to find a balance between sobriety and expressivity ensuring that the mining algorithm is both sober (not overfitting) and precise (not underfitting). Finding an adequate level of generalization is all the more difficult than logs provide only positive instances of behaviour, making it hard to guess which behaviours are not expected to happen.

In the rest of the section, we examine two process mining algorithms. The first algorithm, α, constructs net systems after extracting from logs all ordered pairs of events that appear in sequence in at least one execution sequence in the log. The α-algorithm is sober and very efficient, but its expressivity, not fully characterized, may be limited. The second algorithm, based on regions and the net synthesis techniques presented in this book, has high expressivity, but it is neither sober nor very efficient. Both types of mining algorithms may therefore play complementary roles in practical tools for workflow mining.

3.2.2 The Process Mining Algorithm α

In this section, we present and illustrate the process mining algorithm α [38], and we provide a new characterization of the α-reconstructible workflow nets. The proof of this characterization, given in [1], is not reproduced here (this piece of work is too far from the central topic of this book). In conformity with the general scheme sketched in Sect. 3.2.1, we present the algorithm α as the composition $\alpha = Syn \circ Abs$ of an abstraction function and a synthesis function. The abstraction function Abs is given in the following definition.

Definition 3.21. *The α-abstraction of a workflow log $W \subset T^*$ is the triple $Abs(W) = \langle I_W, C_W, O_W \rangle$ where:*

1. $I_W = \{t \in T \mid \exists u \in T^* \quad t \cdot u \in W\}$ *is the set of transitions starting some execution sequence in the log;*
2. $O_W = \{t \in T \mid \exists u \in T^* \quad u \cdot t \in W\}$ *is the set of transitions ending some execution sequence in the log;*
3. $C_W = \left\{t \cdot t' \in T^2 \mid \exists u, v \in T^* \quad u \cdot t \cdot t' \cdot v \in W\right\}$ *is the set of pairs of transitions appearing consecutively in some execution sequence in the log.*

$\diamondsuit$

The abstraction function Abs is obviously convex (Def. 3.18). When N ranges over workflow nets with set of transitions T, the size of $Abs(N)$ is in $O(|T|^2)$. Moreover, a firing sequence of N contained in a log W may contribute several pairs of transitions in C_W. Therefore, one may expect to find α-complete logs of N with size even smaller than $O(|T|^2)$. In view of these observations, $\alpha = Syn \circ Abs$ is a sober mining algorithm (def. 3.19). In order to complete the description of α, it remains to specify the synthesis function Syn used in this algorithm.

From the theory of event structures [39, 40], we know that the behaviour of a net system can be captured, up to net unfolding, by the basic relations of *causality*, *conflict* and *concurrency* between events. When unfolding a net to an event structure, the events are not in bijective correspondence with the transitions of the net, since two occurrences of the same transition may be distinguished by their past history. Nevertheless, given a workflow net N with set of transitions T and a log W of N, one may derive from the α-abstraction of this log three relations $\rightarrow_W$, $\sharp_W$, $\|_W$ between the transitions of N (the activities reported in the log), approximating loosely the relations of causality, conflict and concurrency in the associated event structure. These relations are the following:

$$\begin{array}{lrcl}
\text{causality:} & t \rightarrow_W t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge \ t' \cdot t \notin C_W \\
\text{conflict:} & t \,\sharp_W\, t' & \Leftrightarrow & t \cdot t' \notin C_W \ \wedge \ t' \cdot t \notin C_W \\
\text{concurrency:} & t \,\|_W\, t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge t' \cdot \ t \in C_W
\end{array}$$

From these relations between transitions, one may derive again the following relations between sets of transitions.

Definition 3.22. *Let $W \subseteq T^*$ be a workflow log. For any sets of transitions $A, B \subseteq T$, let $A \prec_W B$ when the following three conditions hold:*

1. $(\forall a \in A)(\forall b \in B) \quad a \ \rightarrow_W \ b$,
2. $(\forall a_1, a_2 \in A) \quad a_1 \sharp_W a_2$, *and*
3. $(\forall b_1, b_2 \in B) \quad b_1 \sharp_W b_2$

Let $A \prec_W^m B$ when A and B are maximal sets with the property $A \prec_W B$, i.e., $A \prec_W^m B \Leftrightarrow (A \prec_W B) \wedge (A' \prec_W B' \wedge A \subseteq A' \wedge B \subseteq B' \Rightarrow A = A' \wedge B = B')$. ♢

Note that the definition of $\prec_W$ may be applied to any workflow log, and in particular to the full log $\mathcal{L}(N)$ of a workflow net N. Using the above relations, the synthesis function *Syn* used in the α algorithm may be defined as follows.

Definition 3.23. *Let $\langle I_W, C_W, O_W \rangle$ be the α-abstraction of a workflow log $W \subset T^*$. Then $\alpha(W) = Syn(\langle I_W, C_W, O_W \rangle)$ is the (quasi-elementary) net system (P, T, F, M_0) defined as follows:*

1. $P = \{i, o\} \cup \{p_{A,B} \mid A, B \subseteq T \wedge A \prec_W^m B\}$,
2. ${}^\bullet i = \emptyset$, *and* $i^\bullet = I_W$,
3. ${}^\bullet o = O_W$, *and* $o^\bullet = \emptyset$,
4. ${}^\bullet p_{A,B} = A$, *and* ${p_{A,B}}^\bullet = B$,
5. $M_0 = \{i\}$. ♢

We start now a series of illustrative examples. Our first example shows that the synthesis function *Syn* is not injective, even in restriction to α-abstractions of logs of a fixed workflow net N.

Example 3.24. Let N be a worflow net with an initial transition X, a final transition Y, and six concurrent transitions $A, A', A", B, B', B"$ in between. The full log $\mathcal{L}(N)$ of N is the set of all words that start with X, end with Y, and contain every transition of N exactly once. Consider the two logs

$W_1 = \{XA'AA"BB"Y, XB'BB"AA"Y\}$
$W_2 = \{XA'AA"BB"Y, XB'BB"AA"Y, XA'ABB"A"Y, XB'BAA"B"Y\}$

Necessarily $\mathit{Abs}(W_1) \neq \mathit{Abs}(W_2)$. Nevertheless, the nets synthesized by the function *Syn* from $\mathit{Abs}(W_1)$ and $\mathit{Abs}(W_2)$ are both equal to the net depicted in Fig. 3.4. As this net gets deadlocked after A' or B', it is not a workflow

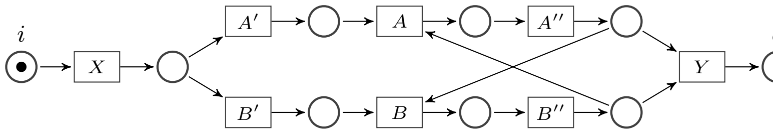


Fig. 3.4. a net system which is not a workflow net

net, hence the α-algorithm will deliver no result in either cases. □

Our second example shows that a workflow net $\alpha(W)$ mined from a workflow log W is not always α-reconstructible.

Example 3.25. Consider the workflow net N_1 depicted in Fig. 3.5. Let $W_1 =$

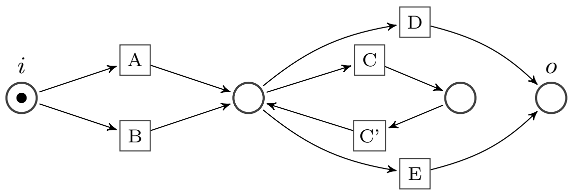


Fig. 3.5. workflow net N_1

$\{ACC'D, BCC'E, AD, BE\}$, then clearly, W_1 is a log of N_1. Applying the α-algorithm to this workflow log produces the workflow net $N_2 = \alpha(W_1)$ shown in Fig. 3.6. The full log of N_2 is the language $\mathcal{L}(N_2) = \{ACC'D, BCC'E\}$.

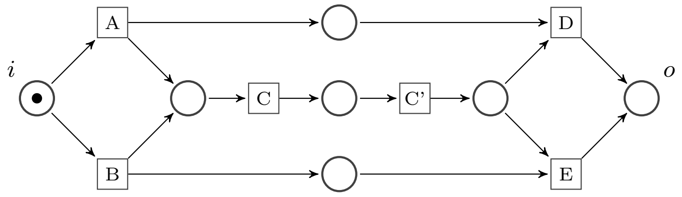


Fig. 3.6. workflow net N_2

Let $W_2 = \mathcal{L}(N_2)$. Applying the α-algorithm to this workflow log produces the workflow net $N_3 = \alpha(W_2)$ depicted in Fig. 3.7. Therefore, $N_2 =$

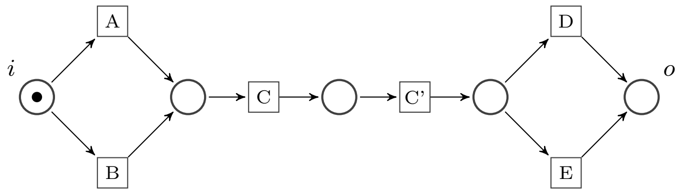


Fig. 3.7. workflow net N_3

$\alpha(W_1)$ is not α-reconstructible. The full log of N_3 is the language $\mathcal{L}(N_3) = \{ACC'D, ACC'E, BCC'D, BCCE\}$. Let $W_3 = \mathcal{L}(N_3)$. Applying the α-algorithm to the workflow log W_3 produces again the workflow net N_3. Therefore, N_3 is α-reconstructible. The neat effect of the *alpha*-algorithm applied to the full log W_2 of N_2 is to remove from this net the two places connected to transitions A and D, resp. B and E. Removing these places from N_2 modifies the language of this net by adding $ACC'E$ and $BCC'D$ as possible execution traces. According to the definitions given below, these places are neither boundary places nor structurally implicit places. □

Definition 3.26. *An inner place p of a workflow net is said to be a* boundary *place when:* $\quad \forall t \in {}^\bullet p \quad \forall t' \in p^\bullet \quad t \cdot t' \in C_{L(N)}$. ♢

Definition 3.27. *A place p of a (contact-free) net system* $N = (P, T, F, M_0)$ *is a* structurally implicit place *if for every reachable marking M and transition* $t \in p^\bullet$, ${}^\bullet t \setminus \{p\} \subseteq M \Rightarrow p \in M$. ♢

It is straightforward to see that if a workflow net N has inner places which are not boundary places, then this workflow net is not α-reconstructible. Worse, if N has inner places which are neither boundary places nor structurally implicit places, then $\mathcal{L}(N) = \mathcal{L}(N')$ for no α-reconstructible workflow net N'. Our last example is intended to illustrate another obstacle to α-reconstructibility, namely the presence of short loops.

Definition 3.28. *Two transitions of a (contact-free) net system form a* short loop *if* $t^\bullet \cap {}^\bullet t' \neq \emptyset$ *and* $t'^\bullet \cap {}^\bullet t \neq \emptyset$. ♢

Example 3.29. The workflow net N of Fig. 3.8 has a short loop involving transitions B and C. A complete log of N is $W = \{ABCBD\}$. The abstraction of this log is $Abs(W) = \{\{A\}, \{AB, BC, CB, BD\}, \{D\}\}$. Since both short sequences BC and CB belong to C_W, one gets $B \parallel_W C$. Thus cyclic dependencies within short loops are lost when applying the α-algorithm. The net system synthesized by the function Syn from $Abs(W)$ is shown in Fig. 3.9. This net it is not a workflow net since the transition C is isolated, hence the α-algorithm produces no result when it is applied to the workflow log W. □

After these examples, let us now state the characterization of α-reconstructible workflow nets established in [1].

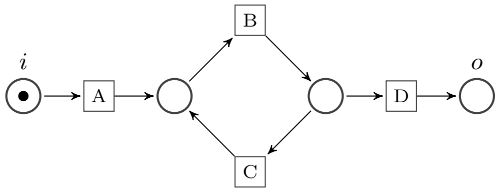


Fig. 3.8. a workflow net with a short loop

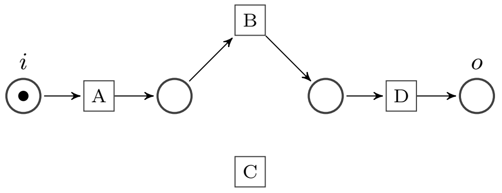


Fig. 3.9. the net system constructed by the algorithm α from the complete log $W = \{ABCBD\}$ of the workflow net of Fig. 3.8

Definition 3.30. *An α-workflow net is a workflow net $N = (P, T, F, M_0)$ in which there are no short loops, all inner places are boundary places, and for any two (non-empty and disjoint) sets of transitions A and B, if $A \prec_W B$ for $W = \mathcal{L}(N)$, then $A \subseteq {}^\bullet p$ and $B \subseteq p^\bullet$ for some place $p \in P$.* $\diamondsuit$

Theorem 3.31. *A workflow net N is isomorphic to $\alpha(\mathcal{L}(N))$, i.e. N is α-reconstructible, if and only if N is an α-workflow net.*

The α-algorithm was originally presented in the context of structured workflow nets. The main result announced in [38] and proven in the report [37] is that structured workflow nets without short loops are α-reconstructible. Let us briefly introduce structured workflow nets. The reader may have observed that the net shown in Fig. 3.6 is not a *free-choice* net [18]. The two choices between events A and B and between events D and E are not independent: if one chooses A (resp. B), then one must choose D (resp. E). In fact, one cannot choose between events D and E at run time, since both events are never jointly enabled. Structured workflow nets satisfy a property slightly stronger that the free-choice property, that already excludes such interferences between conflict (the sharing of an input place by two transitions) and synchronization (the sharing of two input places by a transition).

Definition 3.32. *A workflow net $N = (P, T, F, M_0)$ is a structured workflow net if it has no structurally implicit places and the following condition holds:*

$$\forall t \in T \quad |{}^\bullet t| > 1 \Rightarrow (\forall p \in {}^\bullet t \quad |{}^\bullet p| = 1 \ \wedge \ |p^\bullet| = 1) \tag{SWN}$$

i.e., if a transition t requires the synchronization of several conditions (places), then each of these conditions has a unique cause ($|{}^\bullet p| = 1$) and a unique consequence ($|p^\bullet| = 1$), hence it cannot induce a conflict between t and another transition t'. $\diamondsuit$

The main result established in [37] is the following.

Theorem 3.33. *Structured workflow nets without short loops are α-reconstructible.* □

Th. 3.33 may be established with little effort on the basis of Th. 3.31. The key argument is the observation that, for any pair of transitions t and t' of a structured workflow net, the set $t^\bullet \cap {}^\bullet t'$ contains at most one place. From this restrictive property, many interesting facts follow. In particular, for an inner place, it is equivalent to be a boundary place and not to be a structurally implicit place. This strong property does not hold for general workflow nets. E.g., in the workflow net N_2 shown in Fig. 3.6, the places connected to transitions A and D, resp. B and E are neither boundary places nor structurally implicit places, while in the workflow net considered in Example 3.34 given below, the place p (in grey) is jointly a boundary place and a structurally implicit place. In order to establish Th. 3.33, it suffices then to show that under condition (SWN), if $A \prec_W B$ for $W = \mathcal{L}(N)$, then $A \subseteq {}^\bullet p$ and $B \subseteq p^\bullet$ for some place p of N. Proving this using the fact that $|t^\bullet \cap {}^\bullet t'| \leq 1$ for all transitions t and t' of N is left to the reader as an exercice (see Prob. 3.5 on page 3.5).

Example 3.34. Consider transitions C, F, G in the workflow net N depicted on the left of Fig. 3.10. If we let $W = \mathcal{L}(N)$, then we get $C \sharp_W F$, $C \rightarrow_W G$,

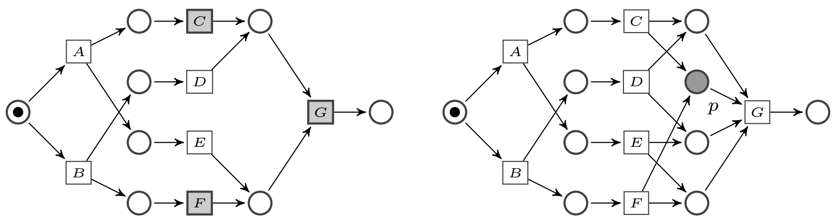


Fig. 3.10. a place p of an α-reconstructible net which is jointly a boundary place and a structurally implicit place

and $F \rightarrow_W G$. Therefore, the α-algorithm necessarily produces a place in $C^\bullet \cap F^\bullet \cap {}^\bullet G$. This is indeed the place p that appears in the net $N' = \alpha(\mathcal{L}(N))$ shown on the right of Fig. 3.10. Another place in $D^\bullet \cap E^\bullet \cap {}^\bullet G$ appears symmetrically in N'. Now N and N' are language equivalent, and therefore, N' is α-reconstructible. It follows from Th. 3.31 that p is a boundary place. However, p is clearly a structurally implicit place of N'. □

The conditions characterizing structured workflow nets (Def. 3.32) are sufficient, but not necessary, to ensure α-reconstructibility. Actually, an α-reconstructible net may contain structurally implicit places (Exple. 3.34) and it may not satisfy condition (SWN) (Exple. 3.8, on page 121).

Adding structurally implicit places to a net preserves its language, and removing places from a net satisfying condition (SWN) cannot invalidate this condition. Therefore, one can state the following corollary to Th. 3.33.

Corollary 3.35. *A workflow net N without short loops and satisfying condition (SWN) is always language equivalent to some α-reconstructible workflow net N'.* $\square$

Condition (SWN) is a structural condition, hence it can be checked very efficiently. It was argued [38] that the class of nets satisfying this condition supports all basic routing patterns and building blocks used to construct workflow systems in practice.

To conclude this section, we note that Example 3.25 provides a language $W_1 = \{ACC'D, BCC'E, AD, BE\}$ with $W_1 \not\subseteq \mathcal{L}(\alpha(W_1)) = \{ACC'D, BCC'E\}$. It follows therefrom that there cannnot exist any order (or pre-order) relation on net systems such that α participates in a Galois connection of the form:

$$W \subseteq \mathcal{L}(N) \quad \Leftrightarrow \quad N \sqsubseteq \alpha(W)$$

In other words, α cannot always provide optimal over-approximations of workflow logs by languages of workflow nets. Recall that when mining a process from a w-language W one should check both that the net system $\alpha(W)$ constructed from W is indeed a workflow net and that W is included in the language of this net system (which is not the case for $W = W_1$).

Summing up, α is an efficient tool for *identifying* from complete logs all workflow nets in the class characterized by Th. 3.31, but it is not targeted to synthesize optimal workflow nets from arbitrary w-languages. In the context of workflow net identification, sobriety is a crucial property, and the α-algorithm enjoys this property. The central result, stated in Theo. 3.33, is the α-reconstructibility of all structured workflow nets without short loops. The subclass of workflow nets without short loops satisfying the condition (SWN) is of special interest since it is characterized by purely structural properties, hence easy to check, and all workflow nets in this subclass are language equivalent to α-reconstructible workflow nets.

3.2.3 A Region-based Mining Algorithm

We have observed that α cannot always provide optimal over-approximations of workflow logs by languages of workflow nets. In this section, we propose another algorithm tailored for this purpose, called ω and based on regions. At start, we apply naively to prefix closures of w-languages the Galois connection established in Section 1.5 between net systems and initialized transition systems (which include prefix closed languages as a particular case). In this way, each w-language W gets associated with the net system $SN(pref(W))$ synthesized from all regions of its prefix closure, but this net is in general not a workflow net. To fix the problem, we specialize regions to workflow-regions

(regions that may appear as extensions of places of workflow nets), and we introduce two specialized net synthesis operations $\omega(W)$ and $\omega_{min}(W)$, producing nets from all workflow regions and from all minimal workflow regions of W, respectively. We show that $\omega(W)$ and $\omega_{min}(W)$ are contact-free net systems and that they are language equivalent. Next, we introduce a condition of ω-completeness of logs W ensuring that $\omega(W)$ and $\omega_{min}(W)$ are workflow nets. We prove that every workflow net is language equivalent to some ω-reconstructible workflow net. We compare the α-algorithm and the ω-algorithm for their respective advantages. We finally propose an incremental version of the ω-algorithm to compensate for its lack of sobriety.

To begin with, we bring back some material scattered in earlier sections. Let $W \subseteq T^*$ be any w-language over T. The prefix closure of W is the language $pref(W) = \{u \in T^* \mid \exists v \in T^* \; u \cdot v \in W\}$. By Def. 3.12, $W = max(pref(W))$ where $max(U) = \{u \in U \mid u \cdot v \in U \Rightarrow v = \varepsilon\}$. Therefore, W and $pref(W)$ determine each other. By Def. 1.83, $pref(W)$ may be considered as an initialized transition system, with words as states. By Th. 1.51, for any net system N with the set of transitions T, the following relation holds:

$$pref(W) \subseteq L(N) \quad \Leftrightarrow \quad N \leq SN(pref(W)) \qquad \text{(GC)}$$

To explain the Galois connection GC, let us recall the following:

1. The prefix-closed language $pref(W)$ is identified with the initialized transition system (S, T, Δ, s_0) defined with $S = pref(W)$, $s_0 = \varepsilon$, and $\Delta = \{(u, t, u \cdot t) \mid u \cdot t \in S\}$.
2. $SN(pref((W))$ is the net system synthesized from all regions of $pref((W))$, i.e., the set of places of this net system is the set $R(pref((W))$.
3. The order relation $\leq$ between net systems is the inclusion between sets of places, where each place p is identified with its signature $p : \{init\} \cup T \to \{-1, 0, 1\}$, i.e., $p(init) = M_0(p)$ and for all $t \in T$, $p(t) = -1$ if $t \in p^\bullet$, $p(t) = 1$ if $t \in {}^\bullet p$, and $p(t) = 0$ otherwise.
4. $L(N) = \{u \in T^* \mid M_0[u\rangle\}$ is the set of all transition sequences that can be fired from the initial marking of N.

Let us examine the special case where the Galois connection is applied to a workflow net N. In this particular case, $L(N) = pref(\mathcal{L}(N))$, since any firing sequence can be extended to a firing sequence that reaches the final marking. As $N \leq N' \Rightarrow L(N') \subseteq L(N)$ and in view of GC, for any $W \subseteq T^*$, the language $L(SN(pref(W)))$ of the net system $SN(pref(W))$ synthezised from the prefix closure of W is the least language of a net system that includes W. However, $SN(pref(W))$ needs not be a workflow net, even if W is a w-language. We will come back to this problem and fix it later on. Ignoring this problem, we would like to propose now a global comparison between α and region-based synthesis algorithms, regarding both the different goals they pursue and the different techniques they use for constructing places of nets.

The main goal of α is the exact reconstruction of processes from sets of execution sequences, and α is particularly good at achieving this goal from complete logs (that need not be full logs). When using α, the working assumption is that the set W of sequences taken as input is a complete log of some workflow net to be reconstructed. Without this assumption, α may behave in an unexpected way, for instance it can produce workflow nets with languages strictly included in W. When using region-based net synthesis algorithms, one does not make any assumption on W. One just tries to construct for all W the simplest net model that contains all sequences in this set. Sobriety, which is crucial to process reconstruction algorithms, has minor importance in this different perspective.

With the α-algorithm, the places of the net $\alpha(W)$, or the pairs $({}^\bullet p, p^\bullet) \in \wp(T) \times \wp(T)$ which represent the places p, are the maximal pairs of sets of transitions (A, B) in the relation $A \prec_W B$ (Def. 3.22). This relation $\prec_W$ is computed solely from the pairs of transitions that appear consecutively in some sequence in W (this *local* information is reported in the component C_W of the α-abstraction of W). With a region based synthesis algorithm, the places p of the synthesized net, or the corresponding pairs $({}^\bullet p, p^\bullet)$, are the regions of W, or the signatures of these regions (for any non-trivial region p, ${}^\circ p$ and p° determine $p(init)$). Now regions cannot be determined from local informations such as given in C_W. On the contrary, regions are determined from constraints induced by arbitrary long factors of all words in W. As a consequence, one cannot expect from region based synthesis algorithm to have the efficiency of α.

We now come back to the problem caused by the fact that $SN(pref(W))$ needs not be a workflow net. In view of relation (GC), the net system $SN(pref(W))$ is maximal w.r.t. net inclusion among net systems N with language $L(N)$ larger than or including $pref(W)$. Consider the particular case where W is a complete log of an α-reconstructible workflow net N. Then, every place of N must also be a place of $SN(pref(W))$, and therefore, $W \subseteq L(SN(pref(W))) \subseteq L(N)$. However, $L(SN(pref(W)))$ may be strictly smaller than $L(N)$, because $SN(pref(W))$ may have more places than N. Owing to the presence of these additional places, $SN(pref(W))$ may even not be a workflow net. The question is then to determine which places of $SN(pref(W))$ should be removed in order to obtain a workflow net. We will address this question for arbitrary workflow logs, i.e. for possibly incomplete logs of possibly not α-reconstructible workflow nets. To fix the problem, we must specialize regions (Def. 1.20) to workflow-regions, i.e., regions that may appear as extensions of places of workflow nets, and consider relativized versions of the net synthesis operator SN restricted to workflow-regions.

Before we define workflow regions, it may be useful to recall the definition of regions of a prefix closed language (Def. 1.20 and Def. 1.83). Given $W \subseteq T^*$, a region of the prefix-closed language $L = pref(W)$ is a subset $r \subseteq L$ such that one of the following cases is met for each transition $t \in T$:

1. $\forall w.t \in pref(W) \quad (w \in r \wedge w.t \notin r) \qquad (t \in r^\circ)$
2. $\forall w.t \in pref(W) \quad (w \notin r \wedge w.t \in r) \qquad (t \in {}^\circ r)$
3. $\forall w.t \in pref(W) \quad (w \in r \wedge w.t \notin r) \qquad (t \in r^\perp)$

By an abuse of terminology, the regions of the language $L = pref(W)$ are also called regions of W.

Example 3.36. Fig. 3.11 shows (on the left) a workflow net N with no inner place, and with just two transitions A and B from the place i to the place o. Let W be the full log of N, thus $W = \{A, B\}$. The prefix-closed language $pref(W)$ defines the transition system shown in the middle of Fig. 3.11. The set of regions of W is $\{\{\varepsilon\}, \{A\}, \{B\}, \{A, B\}\}$. There are three minimal regions: $i = \{\varepsilon\}$ (which represents the extension of the place i), $r_1 = \{A\}$, and $r_2 = \{B\}$. The region $o = \{A, B\}$ (which represents the extension of the place o) is not minimal since it contains the regions r_1 and r_2. The net $SN(pref(W))$ is depicted on the right of Fig.3.11. □

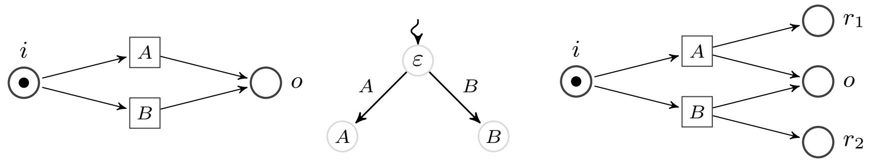


Fig. 3.11. N, $pref(W)$, and the net $SN(pref(W))$

The following remarks give further hints to the definition of workflow regions.

Remark 3.37. If $W \subseteq \mathcal{L}(N)$ is a log of some workflow net N, then it is a w-language (Def. 3.12 on page 122), and $i = \{\varepsilon\}$ is a region of W, with ${}^\bullet i = \emptyset$ and $i^\bullet = I_W$, and $o = W$ is a region of W, with ${}^\bullet o = O_W$ and $o^\bullet = \emptyset$. The region $i = \{\varepsilon\}$ is a minimal region of W, but the region $o = W$ is in general not a minimal region of W. In fact, $o = W$ is not a minimal region as soon as O_W contains at least two transitions of N, because every subset $O \subsetneq O_W$ determines a corresponding region r with ${}^\bullet r = O$ and $r^\bullet = \emptyset$. This situation has already been illustrated in Example 3.36. □

Remark 3.38. Given a w-language W included in the language $\mathcal{L}(N)$ of a workflow net N, let p be an inner place of N. Consider the extension $[\![p]\!]$ of p in W, i.e., the subset of words $u \in pref(W)$ such that $M_0[u\rangle M$ for some marking M of N with $M(p) = 1$. Then $[\![p]\!]$ is a region of W and $[\![p]\!] \cap (\{\varepsilon\} \cup W) = \emptyset$. Indeed, the place p is empty when the place i of N is marked (thus the extensions of p and i cannot intersect), and the place p is empty as well when the place o of N is marked (thus the extensions of p and o cannot intersect). □

We are ready to define (minimal) workflow regions and a relativized version of the net synthesis operator SN.

Definition 3.39. *Given a w-language $W \subseteq T^*$ on T, a workflow-region or ω-region of W is either the region $i = \{\varepsilon\}$, or the region $o = W$, or a region $r \subseteq pref(W)$ (of the transition system $pref(W)$) that intersects neither $i = \{\varepsilon\}$ nor $o = W$. In the latter case, r is called an inner ω-region of W.* $\diamondsuit$

Every region of an initialized transition system is a disjoint union of minimal regions. In particular, every region of a w-language W has a decomposition $r = r_1 \uplus \cdots \uplus r_n$ into minimal regions. Then r is an ω-region if and only if all minimal regions r_i are ω-regions. Therefore, the minimal non-empty ω-regions (w.r.t. set inclusion) are the regions i and o plus the inner ω-regions which coincide with minimal regions.

Notation 3.40 *let $R\omega(W)$ (resp. $Rmin\omega(W)$) denote the set of ω-regions (resp. minimal ω-regions) of a w-language W (more exactly, of its prefix-closure $pref(W)$). Let $\omega(W)$ and $\omega_{\min}(W)$ denote the net systems synthezised from these respective sets of ω-regions, i.e., $\omega(W) = SN_R(pref(W))$ for $R = R\omega(W)$ and $\omega_{\min}(W) = SN_R(pref(W))$ for $R = Rmin\omega(W)$.* $\diamondsuit$

In the rest of the section, we study the two process mining algorithms ω and $\omega_{\min}$ with the synthesis function $Syn = SN$ and with the respective abstraction functions $Abs(W) = R\omega(W)$ and $Abs(W) = Rmin\omega(W)$. We want precisely to show that both algorithms compute optimal over-approximations of workflow logs by workflow nets. In order to reach this objective, one must specialize the Galois connection (GC) between prefix-closed languages and net systems, based on regions of languages, to a similar connection between w-languages and workflow nets, based on ω-regions of w-languages. We propose the adaptation given in the following Def. 3.41 and Prop. 3.43.

Definition 3.41. *A workflow net N with set of transitions T is said to be* compatible *with a w-language $W \subseteq T^*$ on T if $I_W = i^\bullet$ and $O_W = {}^\bullet o$.* $\diamondsuit$

Remark 3.42. Let N be a workflow net with set of transitions T and let $W \subseteq T^*$ be a w-language on T.

1. If $W \subseteq \mathcal{L}(N)$ then N is compatible with W.
2. If N is compatible with W then $\quad W \subseteq L(N) \;\Leftrightarrow\; W \subseteq \mathcal{L}(N)$. $\square$

Proposition 3.43. *If N is a workflow net compatible with a w-language W then $\quad W \subseteq \mathcal{L}(N) \quad \Leftrightarrow \quad N \leq \omega(W)$*

Proof. If N is a workflow net such that $W \subseteq \mathcal{L}(N)$ then, by Remarks 3.37 and 3.38, the extension of every place of N is an ω-region of W, hence $N \leq \omega(W)$. Conversely, if N is a workflow net compatible with W and $N \leq \omega(W)$, then $L(\omega(W)) \subseteq L(N)$. Since the places of the net $\omega(W)$ are regions of W, we moreover have $W \subseteq L(\omega(W))$, and hence $W \subseteq L(N)$. Finally, as N is compatible with W, $W \subseteq \mathcal{L}(N)$ by Rem. 3.42. $\square$

We will now show that for any w-language W, the net systems $\omega(W)$ and $\omega_{min}(W)$ are two language equivalent contact-free net systems. We introduce simplified notations before proving this result (in Prop. 3.47).

Notation 3.44 *Regions $r \in R(A)$ of an initialized system $A = (S, T, \varDelta, s_0)$ are characterized by their signature $r : \{init\} \cup T \to \{-1, 0, 1\}$ (see Rem. 1.52). By an abuse of the notation, we allow ourselves, given two initialized transition systems A_1 and A_2 with the same set of events T, to write $R_1 = R_2$ for sets of regions $R_1 \subseteq R(A_1)$ and $R_2 \subseteq R(A_2)$, respectively, to mean that the regions in R_1 have the same signatures as the regions in R_2.*

Definition 3.45. *Given a w-language W on set of events T, let $\mathcal{T}(W)$ be the initialized transition system $(S, s_0, T, \varDelta)$ defined with $S = (pref(W) \setminus W) \cup \{s_m\}$, $s_0 = \varepsilon$, and $\varDelta = \{(f(w), t, f(w')) \mid w' = w \cdot t \wedge w' \in pref(W)\}$, where $f : pref(W) \to S$ is the map given by $f(u) = u$ for $w \in pref(W) \setminus W$ and $f(u) = s_m$ for $u \in W$.*

Note that $f : pref(W) \to \mathcal{T}(W)$ is the unique label preserving morphism from the transition system $pref(W)$ (see Def. 1.83) to the transition system $\mathcal{T}(W)$, and it is a saturating morphism (Def. 1.72).

Lemma 3.46. *For any w-language W, f^{-1} restricts to a bijection between the minimal ω-regions of W and the minimal regions of $\mathcal{T}(W)$, i.e., $Rmin\omega(W) = Rmin(\mathcal{T}(W))$.*

Proof. The minimal ω-regions $i = \{\varepsilon\}$ and $o = W$ of W are the inverse images $f^{-1}(\{\varepsilon\})$ and $f^{-1}(\{s_m\})$ of the minimal regions $\{\varepsilon\}$ and $\{s_m\}$ of $\mathcal{T}(W)$. Any minimal region of $\mathcal{T}(W)$ that differs from $\{\varepsilon\}$ and $\{s_m\}$ coincides, as a subset of $pref(W)$, with an inner region of W, hence with an ω-region $r = f^{-1}(r)$. Any minimal ω-region r of W that differs from $i = \{\varepsilon\}$ and $o = W$ coincides, as a subset of $pref(W)$, with a region $f(r)$ of $\mathcal{T}(W)$. As f and f^{-1} operate monotonically on regions, the lemma obtains. □

We will now establish the announced statement.

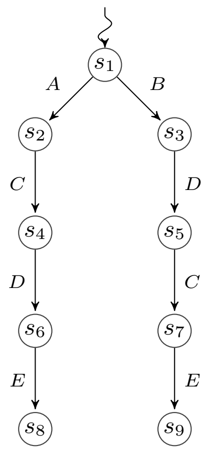
Proposition 3.47. *For any w-language W, $\omega(W)$ and $\omega_{\min}(W)$ are language equivalent net systems: $L(\omega(W)) = L(\omega_{\min}(W)) = L(SN(\mathcal{T}(W)))$. Moreover, they are contact-free.*

Proof. By Prop. 2.18, $SN(\mathcal{T}(W))$ and the net system $SN_R(\mathcal{T}(W))$ defined with $R = Rmin(\mathcal{T}(W))$ have isomorphic reachability graphs. By Lemma 3.46, the regions in $Rmin(\mathcal{T}(W))$ coincide, as maps $r : \{init\} \cup T \to \{-1, 0, 1\}$, with the regions in $Rmin\omega(W)$. In view of this, the net systems $SN_R(\mathcal{T}(W))$ and $SN_{Rmin\omega(W)}(W) = \omega_{\min}(W)$ are isomorphic. By construction, $\omega_{\min}(W) = SN_R(\mathcal{T}(W))$ is contact-free (see Sec. 2.1). As $\omega_{\min}(W)$ is a subnet of $\omega(W)$, $\omega(W)$ is contact-free. In order to complete the proof of the proposition, it suffices to prove that $L(\omega_{\min}(W)) = L(\omega(W))$.

By the proof of Prop. 1.88, a prefix closed language L is of the form $L = L(N)$ for some net system N if and only if the places of N induce regions of L that form an admissible set w.r.t. ESSP. By Prop. 2.18, the net systems $SN(\mathcal{T}(W))$ and $\omega_{\min}(W) = SN_{Rmin(\mathcal{T}(W))}(\mathcal{T}(W))$ have isomorphic reachability graphs, hence they have the same language. Let $L = L(\omega_{\min}(W))$. By the above reasoning, $Rmin\omega(W)$ is an admissible set of regions of L w.r.t. ESSP. Now, $Rmin\omega(W) \subseteq R\omega(W) \subseteq R(\mathcal{T}(W))$, and since $L = L(SN(\mathcal{T}(W)))$, $R\omega(W)$ is also an admissible set of regions of L w.r.t. ESSP, hence $L(\omega_{\min}(W)) = L(\omega(W))$. □

Example 3.48. The set of minimal regions of $W = \{ACDE, BDCE\}$ is the set $Rmin(W) = \{i, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, o\}$ given by $i = \{s_1\}$, $o = \{s_8, s_9\}$, and

$$\begin{aligned}
r_1 &= \{s_2, s_3, s_5\} \\
r_2 &= \{s_2, s_5, s_6, s_8\} \\
r_3 &= \{s_3, s_4, s_7, s_9\} \\
r_4 &= \{s_2, s_3, s_4\} \\
r_5 &= \{s_4, s_6, s_7\} \\
r_6 &= \{s_5, s_6, s_7\} \\
r_7 &= \{s_2, s_4, s_6, s_8\} \\
r_8 &= \{s_3, s_5, s_7, s_9\}
\end{aligned}$$



The net system synthesized from $Rmin(W)$ is shown in the left part of Fig. 3.12. The language of this net system is exactly W. The region r_2 and r_7, resp. r_3 and r_8, are not ω-regions since they contain the terminal state $s_8 = ACDE$, resp. the terminal state $s_9 = BDCE$ but not both.

The ω-regions of W are all regions in $Rmin(W)$ and not in $\{r_2, r_3, r_7, r_8\}$ plus the non-minimal region $r = r_1 \cup r_5 = r_4 \cup r_6 = \{s_2, s_3, s_4, s_5, s_6, s_7\}$. By removing r and $\{r_2, r_3, r_7, r_8\}$, we obtain the net system $\omega_{min}(W) = SN_{Rmin\omega(W)}(W)$ which coincides with $SN_{Rmin(\mathcal{T}(W))}(\mathcal{T}(W))$. We recall that $\mathcal{T}(W)$ is the initialized transition system obtained from the transition system W by fusing the terminal states s_8 and s_9. □

The result stated in Prop. 3.47 is not fully satisfactory, since it may occur that $\omega(W)$ is not a workflow net but there exists a workflow net $N \leq \omega(W)$ such that $W \subseteq \mathcal{L}(N)$. The missing condition for the success of the ω-algorithm is the ω-completeness of logs W, defined as follows.

Definition 3.49. *A w-language $W \subseteq \mathcal{L}(N)$ is said to be an ω-complete log of a workflow net N if $\omega(W) = \omega(\mathcal{L}(N))$, i.e., if $\omega(W) \leq \omega(\mathcal{L}(N))$, or equivalently $R\omega(W) \subseteq R\omega(\mathcal{L}(N))$.* ♢

Def. 3.49 is consistent with the general definition of completeness introduced in Def. 3.16, where one lets $Abs(W) = R\omega(W)$. Note that if a w-language

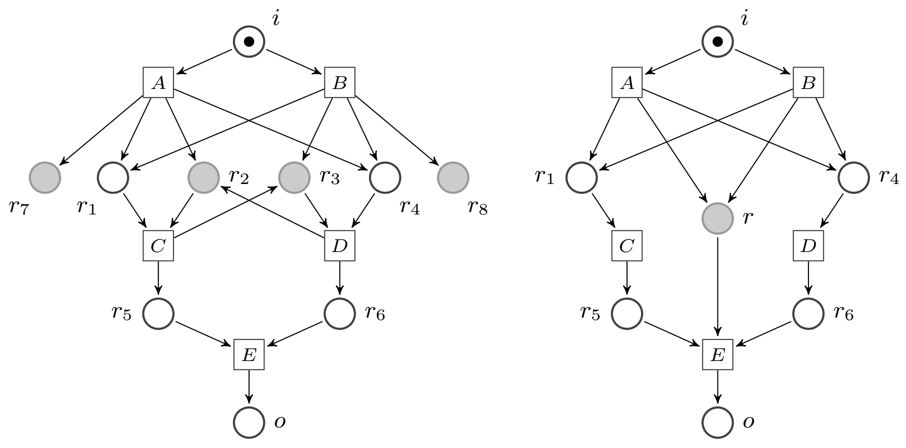


Fig. 3.12. the net system realization of $W = \{ACDE, BDCE\}$ and its version using ω-regions

$W \subseteq \mathcal{L}(N)$ is an ω-complete log of a workflow net N, then W and N are compatible (Def. 3.41).

Proposition 3.50. *The abstraction function $Abs(W) = R\omega(W)$ is convex: if W is an ω-complete log of a workflow net N and if $W \subseteq W' \subseteq \mathcal{L}(N)$, then W' is an ω-complete log of a N.*

Proof. As W is ω-complete, $R\omega(\mathcal{L}(N)) = R\omega(W)$. As $W \subseteq W' \subseteq (\mathcal{L}(N)$, $R\omega(\mathcal{L}(N)) \subseteq R\omega(W') \subseteq R\omega(W)$. Therefore, $R\omega(W') = R\omega(\mathcal{L}(N))$. □

Armed with the condition of ω-completeness, we can now state and prove the main result of the section. The theorem below and its corollaries are not exact counterparts of Theo. 3.31 and Cor. 3.35 for the α-algorithm, and we shall resume the comparison between the α-algorithm and the ω-algorithm after these statements have been established.

Theorem 3.51. *A workflow log W is ω-complete for some workflow net N if and only if $\omega(W)$ is a workflow net, and then W is an ω-complete log of $\omega(W)$ and $\mathcal{L}(\omega(W)) = \mathcal{L}(N)$.*

Proof. Let $W \subseteq T^*$ be a w-language on T. As all places of $\omega(W)$ are regions of W, $W \subseteq L(\omega(W))$. As every transition $t \in T$ occurs in at least one word in W, $\omega(W)$ is initially live.

Assume that W is an ω-complete log of some workflow net N. Then W and N are compatible, and by Prop. 3.43, $N \leq \omega(W)$. N is connected, N and $\omega(W)$ have the same set of transitions, all places that belong to $\omega(W)$ but do not belong to N are inner regions of W, and all inner regions have both a non empty preset and a non empty poset, therefore the net $\omega(W)$ is connected.

Let $t_1 \dots t_n \in \mathcal{L}(\omega(W))$ be a maximal firing sequence of $\omega(W)$ and let M_k, $0 \le k \le n$, be the markings defined by $M_0 = \{i\}$ and $M_{k-1}[t_k\rangle M_k$. By definition of $\mathcal{L}(\omega(W))$, $M_n = \{o\}$. By definition of ω-regions, $i \notin M_k$ for $1 \le k \le n$, and $o \notin M_k$ for $0 \le k < n$.

In order to show that $\omega(W)$ is a workflow net, it remains to check that every firing sequence from the initial marking can be extended to a firing sequence ending in the final marking, i.e., $L_{max}(\omega(W)) = \mathcal{L}(\omega(W))$. Let $u \in L_{max}(\omega(W))$. As $N \le \omega(W) \Rightarrow L(\omega(W)) \subseteq L(N)$ and N is a workflow net, $u \cdot v \in \mathcal{L}(N)$ for some $v \in T^*$. We claim that $\mathcal{L}(N) \subseteq \mathcal{L}(\omega(W))$. Actually, $\mathcal{L}(N) \subseteq \mathcal{L}(\omega(\mathcal{L}(N)))$, and since W is a complete log of N, $\omega(W) = \omega(\mathcal{L}(N))$. Therefore, $u \cdot v \in \mathcal{L}(\omega(W))$. As $u \in L_{max}(\omega(W))$, necessarily, $v = \varepsilon$, and $u \in \mathcal{L}(\omega(W))$ as required. We have thus shown that $\omega(W)$ is a workflow net.

We show now that W is an ω-complete log of $\omega(W)$. By definition of ω-regions, the net system $\omega(W)$ is compatible with W, hence the nets N and $\omega(W)$ have identical postsets $i^\bullet$ and identical presets $^\bullet o$. Therefore, $L(\omega(W)) \subseteq L(N)$ entails $\mathcal{L}(\omega(W)) \subseteq \mathcal{L}(N)$. Since $\mathcal{L}(N) \subseteq \mathcal{L}(\omega(W))$, necessarily $\mathcal{L}(N) = \mathcal{L}(\omega(W))$. As W is an ω-complete log of N, $R\omega(W) = R\omega(\mathcal{L}(N))$, hence $R\omega(W) = R\omega(\mathcal{L}(\omega(W)))$, i.e., W is an ω-complete log of $\omega(W)$.

Supposing now that $\omega(W)$ is a workflow net, we finally show that W is an ω-complete log of $\omega(W)$. By definition of $\omega(W)$, every ω-region of W is a place of this net and hence a region of $L(\omega(W))$. By Prop. 1.75, $L(\omega(W)) \rhd RG(\omega(W))$, hence every region of $RG(\omega(W))$ induces a region of $L(\omega(W))$ with the same signature (Lemma 1.5). Since $\mathcal{L}(\omega(W)) \subseteq L(W)$ any region of $\subseteq L(\omega(W)$ is also a region of $\mathcal{L}(\omega(W))$. Therefore, $R\omega(W) \subseteq R\omega(\mathcal{L}(\omega(W)))$. As $W \subseteq \mathcal{L}(\omega(W))$ entails $R\omega(\mathcal{L}(\omega(W)))$, both sets are equal, hence W is an ω-complete log of $\omega(W)$. □

Corollary 3.52. *Every workflow net N is ω-reconstructible up to language equivalence: $\mathcal{L}(N) = \mathcal{L}(\omega(\mathcal{L}(N)))$, entailing that $\mathcal{L}(N) = \mathcal{L}(\omega(W))$ for any ω-complete log W of N.* □

In view of the above theorem, a workflow log is ω-complete for some workflow net N if and only if it is ω-complete for $\omega(W)$, hence we can speak about ω-complete workflow logs without specifying any reference nets.

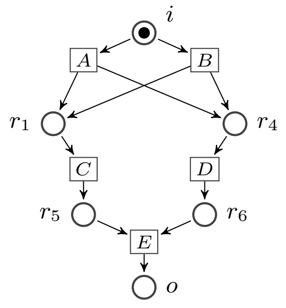
Corollary 3.53. *If a workflow log W is ω-complete, then $L(\omega(W))$ is the least language of a workflow net that contains every execution sequence in W.*

Proof. By Prop.3.43, $\omega(W)$ is the largest workflow net compatible with W, hence $L(\omega(W))$ is the least language of a workflow net that contains every execution sequence in W. □

By Prop. 3.47, $\omega_{\min}(W)$ is a contact-free net system and $\mathcal{L}(\omega_{min}(W)) = \mathcal{L}(\omega(W))$. By reproducing verbatim the arguments given in the proof of Theo. 3.51, one can also show that $\omega_{\min}(W)$ is a workflow net, and that a workflow log W is complete w.r.t. the abstraction function $R\omega$ if and only if it is complete w.r.t. the abstraction function $Rmin\omega$.

Corollary 3.54. *If a w-language $W \subseteq T^*$ is an ω-complete log, then $\omega_{\min}(W)$ is a workflow net and $L(\omega_{\min}(W))$ is the least language of a workflow net that contains W.* □

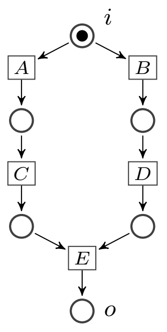
Example 3.55 (Exple. 3.48 continued).



The net system $N = \omega_{\min}(W)$ synthezised from $W = \{ACDE, BDCE\}$ is shown next. N is a workflow net, and

$$\mathcal{L}(N) = \{ACDE, ADCE, BCDE, BDCE\}$$

One may check that $R\omega(W) \subseteq R\omega(\mathcal{L}(N))$, where ω-regions are identified with signatures, hence W is an ω-complete log and $L(N)$ is the least language of a workflow net that contains W.



If we now apply the α-algorithm to the log W, we obtain $A\sharp_W B$, $A\sharp_W D$, $B\sharp_W C$, $C\|_W D$ and the immediate causalities $A \rightarrow_W C$, $B \rightarrow_W D$, $C \rightarrow_W E$, and $D \rightarrow_W E$. The resulting net $\alpha(W)$, given next, is a workflow net but

$$W = \{ACDE, BDCE\} \not\subseteq \mathcal{L}(W) = \{ACE, BDE\}$$

The workflow net depicted on the right in Fig. 3.12 has precisely the language W, hence it is not α-reconstructible.

□

We resume now the comparison between the α-algorithm and the ω-algorithm. The ω-algorithm presents some advantages over ther α-algorithm:

Expressivity Many workflow nets are not α-reconstructible. We have identified a sub-class of α-reconstructible workflow nets (the structured workflow nets) and a related sub-class of workflow nets which are language equivalent to α-reconstructible workflow nets. However, these classes are very restrictive. In contrast, every workflow net is ω-reconstructible.

Approximate realization As shown by Exple. 3.55, α may fail to find a workflow net realizing all computation sequences in a given ω-complete log W, i.e., a workflow net N such that $W \subseteq \mathcal{L}(N)$. In that case, α provides no solution of any kind. In contrast, ω always produces the optimal solution, i.e., a workflow net with the least possible language containing W.

In theory, these are significant advantages. In practice, there are also significant drawbacks, listed below.

Sobriety Even though one can find workflow nets and ω-complete logs thereof which are not α-complete (Exer. 3.7), α-complete logs are often much smaller than ω-complete logs.

Complexity The α-algorithm is much faster and less space consuming than the ω-algorithm.

For algorithm α, sobriety followed from the fact that the abstraction function captures information exclusively from segments of length 2 of words in the log. The notion of region is in contrast not local, e.g., the places p and p' of the workflow net of Fig. 3.13 are not boundary places, but they can be retrieved as regions. The price to pay for increased expressivity is that many execution sequences ought to be present in the log to take such "long distance" dependencies into account.

Example 3.56. Let us consider the two workflow nets depicted in Fig.3.13. N_2

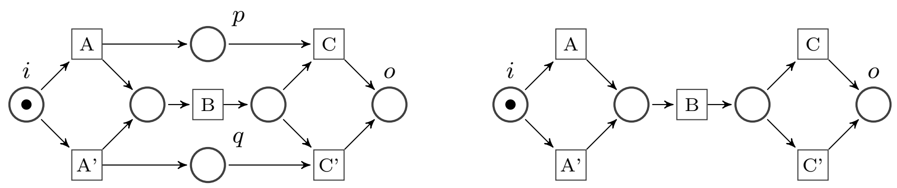


Fig. 3.13. the workflow net N_2 (on the right) constructed by algorithm α from the language of the workflow net N on the left

is the workflow net constructed by algorithm α from $W = \{ABC, A'BC'\}$, i.e., from the language of workflow net N. W is an α-complete log of N_2 but it is not an ω-complete log of N_2. To explain the absence of the place p (respectively of the place q) from N_2, the execution trace ABC' (resp. $A'BC$) should be added to this α-complete log. By doing so, one ends up with the ω-complete log $W_2 = \{ABC, A'BC', ABC', A'BC\}$) of N_2 (which happens to be its full log). Note that W is nevertheless an ω-complete log, since it is the language of a workflow net: $W = \mathcal{L}(N)$. For $k \geq 2$, let us consider the workflow net N_k constructed similarly as N_2 but replacing A and A' (resp. C and C') by k transitions $A_1, \ldots, A_k$ (resp. $C_1, \ldots, C_k$) put in parallel. The set $\{A_iBC_i \mid 1 \leq i \leq k\}$ is an α-complete log of N_k whereas the unique ω-complete log of N_k is the full log $\mathcal{L}(N_k) = \{A_iBC_j \mid 1 \leq i, j \leq k\}$. Indeed every execution sequence A_iBC_j is needed to exclude the (inner) place $p_{i,j}$ such that ${}^\bullet p_{i,j} = \{A_m \mid m \neq i\}$ and $p_{i,j}{}^\bullet = \{C_j\}$. □

To compensate for the lack of sobriety of the ω algorithm, we finally propose in the end of the section an incremental version of the ω algorithm, based on the following observation.

Proposition 3.57. *Let W and W' be two w-languages over T such that $W \subseteq W'$. Then $R\omega(W') = \{r \in R\omega(W) \mid \forall w \in W' \setminus W \quad w \vdash r\}$ where $w \vdash r$*

means that r enables w, *i.e.,* w *belongs to the language of the "atomic" net system* $SN_{\{r\}}(W)$.

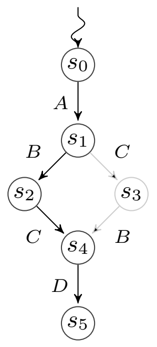
Proof. Left as an exercice (Exer. 3.8). □

In the incremental version of the ω-algorithm, one starts from an initial workflow log W_0 and computes the set of ω-regions of this log (one computes all regions of $\mathcal{T}(W_0)$ and then removes all regions that fail to be ω-regions). Next, whenever new execution sequences are introduced, yielding an increasing sequence of logs $W_0 \subset W_1 \subset \ldots W_{i-1} \subset W_i \ldots$, one removes at each step i every ω-region (of W_0) that does not enable all new execution sequences in $W_i \backslash W_{i-1}$. This yields a decreasing sequence of ω-regions $R_0 \supseteq R_1 \supseteq \ldots R_{i-1} \supseteq R_i \ldots$. By Prop. 3.57, the current net $SN_{R_i}(W_i)$ is always equal to $\omega(W_i)$. In particular, whenever W_i is an ω-complete log, $SN_{R_i}(W_i)$ is an optimal workflow net approximation of W_i.

The incremental version of the ω-algorithm allows saving space. Computing all regions of a log requires storing a full representation of this log, which may be impractical. Difficulties may be avoided by starting with a small log W_0, containing at least one occurrence of every event in T so as to meet the conditions stated in Prop. 3.57. Discarding iteratively from $R\omega(W_0)$ all ω-regions r such that $w \not\vdash r$ for some execution sequence $w \in W_i \setminus W_{i-1}$ does not incur high memory cost.

One may even avoid computing the whole set of regions $R\omega(W_0)$ of the initial log W_0, and compute instead the set of minimal ω-regions $Rmin\omega(W_0)$. Unfortunately, it is not possible, given an increasing sequence of logs $W_0 \subset W_1 \subset \ldots W_{i-1} \subset W_i \ldots$, to compute exactly at each step i the set of minimal ω-regions $Rmin\omega(W_i)$ without storing the current log W_{i-1} or W_i in memory, which is precisely what the incremental version of the ω-algorithm aims at avoiding. The difficulties encountered will be pointed out and discussed after an example.

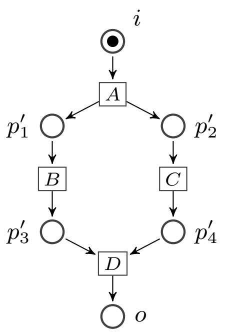
Example 3.58 (See Exer. 3.8). Let $W_0 = \{ABCD\} \subset W_1 = \{ABCD, ACBD\}$.

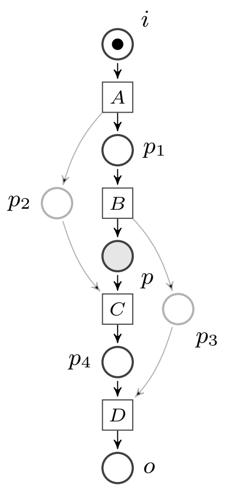


The initialized transition system $\mathcal{T}(W_1)$ is shown on the left. $\mathcal{T}(W_0)$ is the induced restriction of $\mathcal{T}(W_1)$ on the subset of states

$$S = \{s_0, s_1, s_2, s_4, s_5\}$$

The net system $\omega_{\min}(W_1)$, the places of which are the minimal ω-regions of $\mathcal{T}(W_1)$, is shown on the right.





For every region $p'_i \in R(\mathcal{T}(W_1))$, the induced restriction of p'_i on S is a region of $\mathcal{T}(W_0)$ with the same signature, namely $p_i = p'_i \cap S$. Indeed, one has:

$$\begin{array}{ll} p'_1 = \{s_1, s_3\} & p_1 = \{s_1\} \\ p'_2 = \{s_1, s_2\} & p_2 = \{s_1, s_2\} \\ p'_3 = \{s_2, s_4\} & p_3 = \{s_2, s_4\} \\ p'_4 = \{s_3, s_4\} & p_4 = \{s_4\} \end{array}$$

If $r' \in R(\mathcal{T}(W_1))$ and $r \in R(\mathcal{T}(W_0))$ are related regions, i.e., $r = r' \cap S$, then $r \in Rmin\omega(W_0) \Rightarrow r' \in Rmin\omega(W_1)$. However, the converse implication does not hold since p_2 and p_3 are not minimal regions of $\mathcal{T}(W_0)$. Indeed, $Rmin\omega(W_0) = Rmin(\mathcal{T}(W_0)) = \{i, p_1, p, p_4, o\}$, where $p = \{s_2\}$. Moreover, $p = \{s_2\}$ does not coincide with the restriction on S of any region of $\mathcal{T}(W_1)$.
Given the set $W_1 \setminus W_0 = \{ACBD\}$, in order to compute $Rmin\omega(W_1)$ from $Rmin\omega(W_0)$, one may proceed as follows. The unique region in $Rmin\omega(W_0) = Rmin(\mathcal{T}(W_0))$ that disables $ACBD$ is the place p. Therefore, one removes p and replaces it with all regions of $\mathcal{T}(W_0)$ strictly and minimally containing p, namely p_2 and p_3. These regions enable all words in $W_1 \setminus W_0$: $p_2 \vdash ACBD$ and $p_3 \vdash ACBD$. Moreover, $\{i, p'_1, p'_1, p'_3, p'_4, o\}$ are pairwise incomparable regions of $\mathcal{T}(W_1)$. In view of Prop. 3.59 below, $Rmin\omega(W_1)$ is equal to the considered set. □

Proposition 3.59. *Let W and W' be two w-languages over T such that $W \subseteq W'$. Then $Rmin\omega(W') = min_{W'}\{r \in R\omega(W) \mid \forall w \in W' \setminus W \quad w \vdash r\}$ where $min_{W'}$ denotes the operation on sets of regions $r \in R\omega(W)$, identified by their signatures with regions $r \in R\omega(W')$, that extracts from a set the minimal elements of this set w.r.t. the inclusion of sets of states of $\mathcal{T}(W')$.*

Proof. Left as an exercice (Exer. 3.8). □

A naive algorithm based on Prop. 3.59 would consist of computing initially $Rmin\omega(W_0)$ and proceeding as follows with the increasing sequence of logs $W_0 \subset W_1 \subset \ldots W_{i-1} \subset W_i \ldots$. At step i, one removes all regions $r \in Rmin\omega(W_{i-1})$ that do not enable all new sequences in $W_i \setminus W_{i-1}$. Any such region $r \in Rmin\omega(W_{i-1})$, identified by its signature with a corresponding region $r \in R\omega(W_0)$, is then replaced with the set

$$min_{W_0}\{r' \in R\omega(W_0) \mid r \subset r' \wedge \forall w \in W_i \setminus W_{i-1} \;\; w \vdash r\}$$

Let $R_i \subseteq Rmin\omega(W_i)$ be the set of regions obtained after these replacements have been done, then it remains to compute $Rmin\omega(W_i) = min_{W_i}(R_i)$. Unfortunately, W_0 and $W_i \setminus W_{i-1}$ do not provide enough data for computing $min_{W_i}(R_i)$ from the signatures of the regions in R_i, and one needs using explicitly the whole log W_i for this purpose. This should be avoided in an incremental algorithm, hence we propose to avoid computing $min_{W_i}(R_i)$ and use instead the set R_i. The net synthesized from R_i is a representation of the equivalent but smaller net $\omega_{min}(W_i)$.

According to this final algorithm, one explores uniquely the set of ω-regions $R\omega(W_0)$ of the initial log. The successive increments $W_i \setminus W_{i-1}$ serve only to filter out regions from this set. To compute $Rmin\omega(W_0)$, or a "small" set of ω-regions containing $Rmin\omega(W_0)$, one can apply the general algorithm presented in Sections 2.3.1 to 2.3.2, where the refinement process of rough regions is stopped at all terminal nodes $\langle Y_\bullet, Y_\circ \rangle$ (of the trees $T(X_k)$), yielding regions $r = Y_\bullet$. Let R_0 be the resulting set of regions, thus $Rmin\omega(W_0) \subseteq R_0$. Whenever some region $r \in R_0$ is removed because it does not enable all sequences in $W_1 \setminus W_0$, to compute a "small" set of ω-regions enabling $W_1 \setminus W_0$ and strictly containing r, it suffices to resume the refinement process of rough regions from the terminal node $\langle Y_\bullet, Y_\circ \rangle$ that produced $r = Y_\bullet$. The new terminal nodes $\langle Z_\bullet, Z_\circ \rangle$ obtained from this refinement actually yield regions $r' = Z_\bullet$ larger than r and none of them is missed. We iterate the same refinement process from these new terminal nodes $\langle Z_\bullet, Z_\circ \rangle$ such that the region $Z_\bullet$ does not enabled all sequences in $W_1 \setminus W_0$. The same technique may be applied at all steps to compute R_i from the set of nodes $\langle Y_\bullet, Y_\circ \rangle$ (of the trees $T(X_k)$) that determine regions $r = Y_\bullet \in R_{i-1}$.

Problems

3.1. Let N' be the net obtained by gluing the input and output places of a workflow net N (Def. 3.9 on p.122).
(a) Show that $\forall t \in T \quad {}^\bullet t \neq \emptyset \ \wedge \ t^\bullet \neq \emptyset$.
(b) Show that N' is strongly connected.
(c) Show the following:

1. $i^\bullet = \{t \in T \mid \exists u \in T^* \quad t \cdot u \in \mathcal{L}(N)\}$.
2. ${}^\bullet o = \{t \in T \mid \exists u \in T^* \quad u \cdot t \in \mathcal{L}(N)\}$.
3. $(\forall p \in P \setminus \{i, o\}) \quad {}^\bullet p \cap {}^\bullet o = \emptyset \ \wedge \ p^\bullet \cap i^\bullet = \emptyset$.

3.2. Given a (quasi-elementary) net system N, let W be a set of firing sequences of N. In order to allow the reconstruction of N from the α-abstraction $Abs(W)$, given in Def. 3.21, the relations $\rightarrow_W$, $\sharp_W$ and $\|_W$ defined respectively as:

$$\begin{array}{lrcl} \text{causality:} & t \rightarrow_W t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge \ t' \cdot t \notin C_W \\ \text{conflict:} & t \,\sharp_W\, t' & \Leftrightarrow & t \cdot t' \notin C_W \ \wedge \ t' \cdot t \notin C_W \\ \text{concurrency:} & t \,\|_W\, t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge \ t' \cdot t \in C_W \end{array}$$

should fit at best with the corresponding relations defined respectively as:

$$\begin{array}{rcl} t \rightarrow_N t' & \Leftrightarrow & t^\bullet \cap {}^\bullet t' \neq \emptyset \\ t \,\sharp_N\, t' & \Leftrightarrow & ({}^\bullet t \cap {}^\bullet t') \cup (t^\bullet \cap t'^\bullet) \neq \emptyset \\ t \,\|_N\, t' & \Leftrightarrow & ({}^\bullet t \cup t'^\bullet) \cap ({}^\bullet t \cup t'^\bullet) = \emptyset \end{array}$$

For any marking M and for any a sequence of transitions $u \in T^*$ enabled in M, let $M[u\rangle M_u$.
(a) Show that

$$(M[t \cdot t'\rangle \ \wedge \ M[t'\rangle) \ \Leftrightarrow \ (t \|_N t' \ \wedge \ {}^\bullet t \cup {}^\bullet t' \subseteq M \ \wedge \ M \cap (t^\bullet \cup t'^\bullet) = \emptyset)$$

and that moreover $M[t' \cdot t\rangle$ and $M_{t \cdot t'} = M_{t' \cdot t}$ in this case.
(b) Show that

$$M[t \cdot t'\rangle \ \Leftrightarrow \ \neg(t \sharp_N t') \ \wedge \ {}^\bullet t \cup ({}^\bullet t' \setminus t^\bullet) \subseteq M \ \wedge \ M \cap (t^\bullet \cup (t'^\bullet \setminus {}^\bullet t) = \emptyset)$$

(c) Show that, if $M[t \cdot t'\rangle$ and $t^\bullet \cap {}^\bullet t' = {}^\bullet t \cap t'^\bullet = \emptyset$, then $M[t'\rangle$ and hence $t \|_N t'$. Show that, if N is contact-free, then

$$(M[t \cdot t'\rangle \ \wedge \ t^\bullet \cap {}^\bullet t' = \emptyset) \ \Rightarrow \ M[t'\rangle$$

(d) Show that $t \sharp_N t' \Rightarrow t \sharp_W t'$, and that $t \sharp_N t' \Leftrightarrow t \sharp_W t'$ if t and t' are co-enabled, i.e., $M[t\rangle$ and $M[t'\rangle$ for some reachable marking M.
(e) Show that the following relations hold if N is contact-free:

1. $t \rightarrow_W t' \Rightarrow t \rightarrow_N t'$,
2. $t \|_W t' \Rightarrow t \|_N t'$,
3. if t and t' are co-enabled then $t \|_N t' \Leftrightarrow t \|_W t'$.

(f) Show that $\rightarrow_{\mathcal{L}(N)} = \rightarrow_N$ if N is a workflow net without short loops (i.e., $t^\bullet \cap {}^\bullet t' \neq \emptyset$ and $t'^\bullet \cap {}^\bullet t \neq \emptyset$ are mutually exclusive) and all inner places of N are boundary places.

3.3. Given a workflow net $N = (P, T, F, \{i\})$ and two concurrent transitions t and t' of this net ($t \|_N t'$), if $\sigma = u \cdot t \cdot t' \cdot v$ is an execution sequence of N, then in view of (b) in Prob. 3.2, $u \cdot t' \cdot t \cdot v$ is also an execution sequence of N. Therefore, the language of N is closed under the congruence $\sim$ generated by the relations $t \cdot t' \sim t' \cdot t$ pertaining to pairs of concurrent transitions t and t'. An equivalence class of maximal execution sequences of N is called a *process* of N. Processes of a workflow net N may be represented equivalently as pairs $\mathcal{R} = (R, \ell)$ consisting of a net $R = (P_R, T_R, F_R)$ and two labelling functions $\ell : T_R \rightarrow T$ and $\ell : P_R \rightarrow \wp(P)$ satisfying the following conditions:

1. There is a unique place i_R such that ${}^\bullet i_R = \emptyset$, and $\ell(i_R) = \{i\}$ where i is the input place of the workflow net N.
2. There is a unique place o_R such that ${o_R}^\bullet = \emptyset$, and $\ell(o_R) = \{o\}$ where o is the output place of the workflow net N.
3. Every place has at most one incoming arc and one outgoing arc:

$$\forall p \in P_R \qquad |{}^\bullet p| \leq 1 \quad \text{and} \quad |p^\bullet| \leq 1$$

4. The underlying graph of R is acyclic.
5. If $p_R \in {t_R}^\bullet \cap {}^\bullet t'_R$ then $\ell(p_R) = \ell(t_R)^\bullet \cap {}^\bullet \ell(t'_R)$

6. $\{\ell(p_R) \mid p_R \in {}^\bullet t_R\}$ is a partition of ${}^\bullet\ell(t_R)$.
7. $\{\ell(p_R) \mid p_R \in t_R{}^\bullet\}$ is a partition of $\ell(t_R)^\bullet$.

The above definition differs slightly from the usual definition of processes as occurrence nets. The sole difference is that here, *each place in a process R is mapped by ℓ to a set of places of N, playing indistinguishable roles in this process.* As a result, processes are free from equivalent places. In the sequel, we let $\ell(M_R) = \bigcup\{\ell(p_R) \mid p_R \in M_R\}$ denote the marking of N associated by ℓ with the marking M_R of R.

(a) Show that for any reachable marking M_R of R, $M_R[t_R\rangle M'_R$ in R if and only if $\ell(M_R)[\ell(t_R)\rangle\ell(M'_R)$ in N.
(b) Show that R is a workflow net and show that, if N has no short loops, then R has no short loops.
(c) Show that every transition in T_R occurs exactly once in any maximal execution sequence of R.
(d) Show that the set $\mathcal{L}_\mathcal{R} = \{\ell(t_1)\cdots\ell(t_n) \mid t_1\cdots t_n \in \mathcal{L}(R)\}$ is an equivalence class of maximal execution sequences of N w.r.t. the permutation equivalence $\sim$ (we recall that $\mathcal{L}(R)$ is the set of maximal firing sequences of the workflow net R, thus starting in $\{i_R\}$ and ending in $\{o_R\}$).
(e) Construct for each maximal execution sequence σ of N an associated process $\mathcal{R} = (R, \ell)$ of N such that $\sigma \in \mathcal{L}_\mathcal{R}$.
(f) Show that a place $p_R \in (t_R{}^\bullet \cap {}^\bullet t'_R)$ is structurally implicit in R if and only if there does not exist any execution sequence in $\mathcal{L}(R)$ in which transition t_R is immediately followed by t'_R.
(g) Show that the complete processes $\mathcal{R} = (R, \ell)$ of a workflow net N are in bijective correspondence with the equivalences classes of the maximal execution sequences of N modulo $\sim$, where the correspondence is given by $\mathcal{L}_\mathcal{R} = \{\ell(t_1)\cdots\ell(t_n) \mid t_1\cdots t_n \in \mathcal{L}(R)\}$.
(h) Show that an inner place p of a workflow net is a boundary place if and only if for every pair of transitions $t \in {}^\bullet p$ and $t' \in p^\bullet$, there exists a process $\mathcal{R} = (R, \ell)$ of N and a **non structurally implicit** place $p_R \in (t_R{}^\bullet \cap {}^\bullet t'_R)$ in this process such that $\ell(t_R) = t$ and $\ell(t'_R) = t'$ (hence $p \in \ell(p_R)$).
(i) Show that the workflow net depicted in Fig. 3.3 on page 3.3 has two processes. Infer from this that all inner places of this workflow net are boundary places.
(j) Show that the workflow net depicted in Fig. 3.6 on page 3.6 has two complete processes. Infer from this that all inner places of this workflow net are boundary places except for $p \in A^\bullet \cap {}^\bullet D$ and $q \in B^\bullet \cap {}^\bullet E$.

3.4. This exercise aims at providing a comparison between the condition (SWN) defined on page 130 with the free-choice condition on nets. A net is *N-free* if its underlying graph does not contain any subgraph consisting of two places p and q and two transitions t and u such that p is simultaneously a pre-place of t and u, and t is simultaneously a post-transition of p and q, i.e. they form

an "N" pattern $\begin{tikzcd} & p \arrow[dl] \arrow[dr] & & q \arrow[dl] \\ \boxed{u} & & \boxed{t} & \end{tikzcd}$. The condition of N-freeness stipulates that for every transition t and for every pre-place $p \in {}^\bullet t$, either t is the unique post transition of p, or p is the unique pre-place of t. In other words, p cannot be involved jointly in a synchronization (with another pre-place of t) and in a choice (between t and another post-transition). A net is *free-choice* if it is contact-free and every pair of transitions sharing some pre-place have an identical set of pre-places: ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset \;\Rightarrow\; {}^\bullet t_1 = {}^\bullet t_2$. The choice is "free" because any pair of conflicting transitions t_1 and t_2 enabled jointly in some marking M ($M[t_1\rangle$ and $M[t_2\rangle$ but neither $M[t_1 \, . \, t_2\rangle$ nor $M[t_2 \, . \, t_1\rangle$) are always jointly enabled or jointly disabled in any marking. Note that in this case, necessarily ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ since the net is contact-free.
(a) Show that a net which satisfies the condition (SWN) is N-free, and that a net which satisfies the condition N-free is free-choice.
(b) Show that a net is contact-free if and only if it can be assembled from *clusters* defined as follows. A cluster is a net (P, T, F) such that $F = P \times T$, see Fig. 3.14 (left). For assembling a number of clusters, one takes their disjoint union and one adds arcs from transitions of a cluster to place of other clusters), but one cannot add any arc from places to transitions.
(c) Show that a net satisfies the condition N-free if and only if it can be

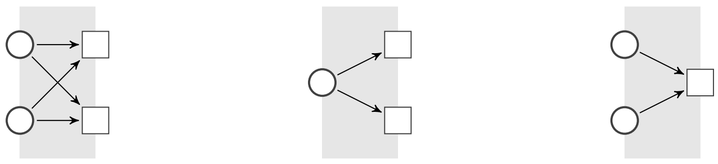


Fig. 3.14. the general form of a cluster (left), and two specific cases: for choice (center) and synchronization (right)

assembled from clusters (P, T, F) in which at least one of the sets P or T is a singleton, see Fig. 3.14 (center and right).
(d) What condition should be added to the assembly rule to obtain exactly the set of nets satisfying condition (SWN)?

3.5. The purpose of this problem is to show that structured workflow nets (Def. 3.32, on p. 130) without short loops are α-reconstructible.
(a) Show that, if t and t' are transitions of a workflow net satisfying the condition (SWN), then $t^\bullet \cap {}^\bullet t'$ contains at most one place.
(b) Show that an inner place of a workflow net satisfying the condition (SWN) is a boundary place if and only if it is not a structurally implicit place.
(c) Show that in a structured workflow net, the following relations are satisfied:

$$(t_1 \rightarrow_N t \;\wedge\; t_2 \rightarrow_N t \;\wedge\; t_1 \sharp_{\mathcal{L}(N)} t_2) \Rightarrow t_1 \sharp_N t_2$$
$$(t \rightarrow_N t_1 \;\wedge\; t \rightarrow_N t_2 \;\wedge\; t_1 \sharp_{\mathcal{L}(N)} t_2) \Rightarrow t_1 \sharp_N t_2$$

(d) Conclude that a structured workflow net without short loops is α-reconstructible.

3.6. Construct the workflow nets $\omega_{\min}(W)$ synthezised from the full logs $W = \mathcal{L}(N)$ of the workflow nets N depicted in Fig. 3.3 and Fig. 3.6, and for the workflow net N defined in Exple. 3.34.

3.7 ([10]). Check that the workflow net depicted in Fig. 3.15 is α-reconstructible.

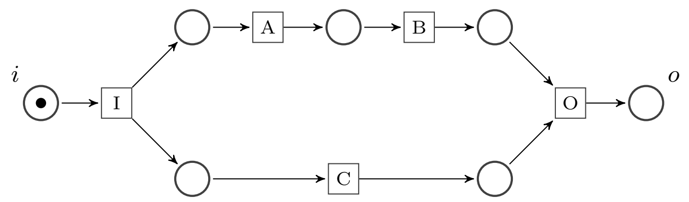


Fig. 3.15. a workflow net

Check that the full log of this net is the unique α-complete log. Show that this workflow net can be synthesized from a strictly smaller log by the ω-algorithm.

3.8. Let $A = (S, E, \varDelta, s_0)$ be an initialized transition system.
(a) Show that $L(SN(A)) = \bigcap \left\{ L\left(SN_{\{r\}}(A)\right) \mid r \in R(A) \right\}$
$= \bigcap \left\{ L\left(SN_{\{r\}}(A)\right) \mid r \in Rmin(A) \right\}$ (hint: use Prop. 2.18).
(b) Assume that A is the induced restriction on a subset of states $S \subseteq S'$ of another initialized transition system $A' = (S', E, \varDelta', s_0)$, i.e., $\varDelta = \varDelta' \cap S \times E \times S$. Note that A and A' have the same set of events E, and since we consider only reduced transition systems (Def. 1.30), every event $e \in E$ occurs in $L(A)$ and in $L(A')$. Show that the restriction $r = r' \cap S$ of a region $r' \in R(A')$ is a region of A with the same signature, and hence that the map $r' \mapsto r' \cap S$ is injective, i.e., $R(A')$ can be seen as a subset of $R(A)$.
(c) Let $r' \in R(A')$ and $r = r' \cap S \in R(A)$. Show that *(i)* $r \in Rmin(A) \Rightarrow r' \in Rmin(A')$, but *(ii)* possibly $r' \in Rmin(A')$ and $r \notin Rmin(A)$.
(d) Let W and W' be two w-languages over T such that $W \subseteq W'$. Show that $R\omega(W') = \{r \in R\omega(W) \mid \forall w \in W' \setminus W \quad w \vdash r\}$ where $w \vdash r$ means that r enables w, i.e., w belongs to the language of the "atomic" net system $SN_{\{r\}}(W)$.
(e) Show that $Rmin\omega(W') \neq \{r \in Rmin\omega(W) \mid \forall w \in W' \setminus W \quad w \vdash r\}$ in the general case (hint: use Example 3.8).
(f) Show that $Rmin\omega(W') = min_{W'} \{r \in R\omega(W) \mid \forall w \in W' \setminus W \quad w \vdash r\}$ where $min_{W'}$ denotes the operation on sets of regions $r \in R\omega(W)$, identified by their signatures with regions $r \in R\omega(W')$, that extracts from a set the minimal elements of this set w.r.t. the inclusion of sets of states of $\mathcal{T}(W')$.

Conclusion

References

1. Eric Badouel. The α and the ω of process mining., December 2011. Inria Research Report.
2. Eric Badouel, Luca Bernardinello, and Philippe Darondeau. The synthesis problem for elementary net systems is NP-complete. *Theor. Comput. Sci.*, 186(1-2):107–134, 1997.
3. Eric Badouel and Philippe Darondeau. Dualities between nets and automata induced by schizophrenic objects. In David H. Pitt, David E. Rydeheard, and Peter Johnstone, editors, *Category Theory and Computer Science*, volume 953 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 1995.
4. Marek Bednarczyk and Philippe Darondeau. Looking for Diamonds. In *Synthesis and Control of Discrete Event Systems*, pages 213–221. Kluwer, 2002.
5. Luca Bernardinello. Synthesis of net systems. In Marco Ajmone Marsan, editor, *Application and Theory of Petri Nets*, volume 691 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1993.
6. Luca Bernardinello, Carlo Ferigato, and Lucia Pomello. An algebraic model of observable properties in distributed systems. *Theor. Comput. Sci.*, 290(1):637–668, 2003.
7. Andrzej M. Borzyszkowski and Philippe Darondeau. Transition systems without transitions. *Theor. Comput. Sci.*, 338(1-3):1–16, 2005.
8. Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors. *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 254 of *Lecture Notes in Computer Science*. Springer, 1987.
9. Nadia Busi and G. Michele Pinna. Characterizing workflow nets using regions. In *SYNASC*, pages 399–406. IEEE Computer Society, 2006.
10. Nadia Busi and G. Michele Pinna. Process discovery and petri nets. *Mathematical Structures in Computer Science*, 19(6):1091–1124, 2009.
11. Josep Carmona, Jordi Cortadella, and Enric Pastor. A structural encoding technique for the synthesis of asynchronous circuits. *Fundam. Inform.*, 50(2):135–154, 2002.
12. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and A. Yakovlev. Complete state encoding based on the theory of regions. In *In International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu*, pages 36–47. Society Press, 1996.

13. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. A region-based theory for state assignment in speed-independent circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 16(8):793–812, 1997.
14. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer Series in Advanced Microelectronics. Springer, 2002.
15. Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Enric Pastor, Oriol Roig, and Alexandre Yakovlev. Checking signal transition graph implementability by symbolic BDD traversal. In *Proc. of the European Design and Test Conference (EDTC'95)*. IEEE Computer Society, 1995.
16. Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alexandre Yakovlev. Deriving Petri nets for finite transition systems. Research Report UPC-DAC-1996-19, Univ. Polytech. Catalunya, 1996.
17. Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alexandre Yakovlev. Deriving Petri nets for finite transition systems. *IEEE Trans. Computers*, 47(8):859–882, 1998.
18. Jörg Desel and Javier Esparza. *Free Choice petri Nets*. Cambridge Tracts in Theoretical Computer Science, 40. Cambridge University Press, 1995.
19. Jörg Desel and Wolfgang Reisig. The synthesis problem of petri nets. In Patrice Enjalbert, Alain Finkel, and Klaus W. Wagner, editors, *STACS*, volume 665 of *Lecture Notes in Computer Science*, pages 120–129. Springer, 1993.
20. Jörg Desel and Wolfgang Reisig. The synthesis problem of petri nets. *Acta Inf.*, 33(4):297–315, 1996.
21. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. part i: Basic notions and the representation problem. *Acta Inf.*, 27(4):315–342, 1989.
22. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. part ii: State spaces of concurrent systems. *Acta Inf.*, 27(4):343–368, 1989.
23. Javier Esparza, Petr Jancar, and Alexander Miller. On the complexity of consistency and complete state coding for signal transition graphs. In *ACSD*, pages 47–56. IEEE Computer Society, 2006.
24. Michael R. Garey and David S.Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
25. Kunihiko Hiraishi. Some complexity results on transition systems and elementary net systems. *Theor. Comput. Sci.*, 135(2):361–376, 1994.
26. C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.
27. Victor Khomenko. Efficient automatic resolution of encoding conflicts using STG unfoldings. In *IEEE transactions on VSLI Systems: special section on Asynchronous Circuits and System*, volume 17, July 2009.
28. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theor. Comput. Sci.*, 96(1):3–33, 1992.
29. Zdzislaw Pawlak. Rough sets. *International Journal of Parallel Programming*, 11(5):341–356, 1982.
30. Dominique Perrin. Finite automata. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 1–57. 1990.
31. Grzegorz Rozenberg. Behaviour of elementary net systems. In Brauer et al. [8], pages 60–94.

32. Grzegorz Rozenberg and Joost Engelfriet. Elementary net systems. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 12–121. Springer, 1998.
33. Grzegorz Rozenberg and P. S. Thiagarajan. Petri nets: Basic notions, structure, behaviour. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 585–668. Springer, 1986.
34. P.S. Thiagarajan. Elementary net systems. In Brauer et al. [8], pages 26–59.
35. Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
36. Wil M. P. van der Aalst, Vladimir Rubin, H. M. W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and Christian W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and System Modeling*, 9(1):87–111, 2010.
37. Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Which processes can be rediscovered?, 2002. BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven.
38. Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
39. Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.
40. Glynn Winskel. An introduction to event structures. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *Lecture Notes in Computer Science*, pages 364–397. Springer, 1988.

Index