TUM

INSTITUT FÜR INFORMATIK

A Compositional Partial Order Semantics for Petri Net Components

Ekkart Kindler

TUM-I9613
SFB-Bereich: 342
März 96

TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM-INFO-03-96-I9613-200/1.-FI

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

A Compositional Partial Order Semantics for Petri Net Components

Ekkart Kindler*
Humboldt-Universität zu Berlin, Institut für Informatik, D-10099 Berlin, Germany

Abstract

In this paper we introduce the concept of a Petri net component and show how systems can be composed from components. A component communicates with its environment via distinguished input and output places, which formalizes communication by message passing.

Then, we present a compositional semantics for components. The semantics is an extension of processes for place/transition systems (partial order semantics). We show that the semantics is fully abstract with respect to the behaviour of closed components (essentially, processes of place/transition systems). A main feature of the compositional semantics is that composition of components corresponds to conjunction.

The semantics can be easily combined with a temporal logic interpreted on processes of a place/transition system. We introduce such a logic and show how it can be combined with the semantics to achieve a rely-guarantee concept.

Keywords: Petri net component, compositional semantics, rely-guarantee specification, partial order semantics, fully abstract.

Introduction

For a modular design of distributed systems a system must be built from components. Furthermore, it must be easy to derive the behaviour of a composed system from the behaviour of its components. For that reason *compositional semantics* are essential. A compositional semantics, however, is not of great use if it cannot be dealt with in an easy way. To this end, the *rely-guarantee-style* for specifying a component has been propagated [Jon83]: the behaviour of a component is specified in dependency of the behaviour of its environment.

In this paper, we introduce the concept of *Petri net components* — a Petri net equipped with *input* and *output* places. Components can be composed at these input and output places. We present a compositional semantics for components and show how components

---

*Correspond to: `kindler@informatik.hu-berlin.de`

can be specified in a rely-guarantee-style by help of a temporal logic.

The key idea of the compositional semantics is known from other formalisms [BKP84, Pnu85, AL90, Pan90]: The semantics is a set of runs; in a run of a component we represent transitions of the component as well as possible transitions of its environment. Since we use *partial order semantics* (processes of place/transition nets [GR83]), we can avoid some technical problems occurring in interleaving semantics [AL90]. Therefore, this semantics is particularly appealing — from a theoretical as well as from an application point of view. Though Petri nets are often reputed to be not compositional, there already exists a compositional partial order semantics (and some more compositional interleaving semantics), which was introduced by Mazurkiewicz [Maz88]. Mazurkiewicz' semantics is based on fusion of transitions, which corresponds to synchronous communication (like in CCS). In contrast, we combine components at input and output places, which corresponds to asynchronous communication via message passing.

For the specification of components we introduce a temporal logic, which is interpreted on runs. It is similar to the Petri net logics introduced in [Rei92, GK+93, KW95, Rei95]; but we allow runs which are not 1-safe, too.

1 Introductory example

First, we present a simple example of a Petri net *component*. This example will help to understand the notion of a component and the *composition* of components. Moreover, we motivate the particular choice of our semantics by help of this example.

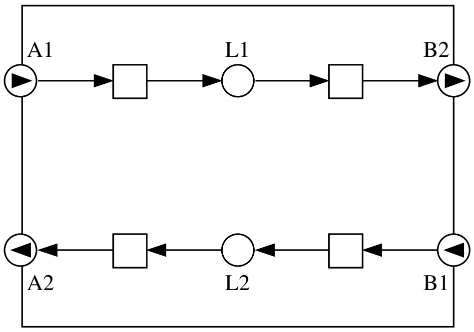

Figure 1: A channel

Figure 1 shows a Petri net surrounded by a box. Some places are arranged at the border of the surrounding box and are marked by an arrowhead. These places represent the *interface* of the component to its environment. An arrowhead pointing into the box indicates an *input place* of the component on which the environment may put some tokens; the environment may not remove a token from an input place. An arrowhead in the opposite direction indicates an *output place* of the component; the environment is allowed to remove a token from an output place, but may not put a token on it. The component itself may remove tokens from input places and put tokens to output places, but not vice versa. The component of Fig. 1 models a simple bidirectional *channel* between two sites, which we call A and B.

Figure 2 models the behaviour of the two sites A and B. Site A models a producer and site B models a consumer.

Figure 2: A producer- and a consumer-component

Now, we can *compose* these three components at the corresponding interface places (places with the same name). The composition is shown in Fig 3. In the composition the combined interface places become *internal places*, which are initially unmarked. When two components are composed it may happen, that an interface place of one component has no corresponding interface place in the other component; then, this place will be an interface place in the combined system. In our example, however, there are no interface places in the composed system any more; therefore, we call it a *closed component.* Since the interface places are supposed to model communication by message passing, we do only combine input places of one component with output places of the other (and vice versa). Moreover, an internal place of one component is not allowed to occur in the other component. When two components are not composable for this reason, they must be renamed before.

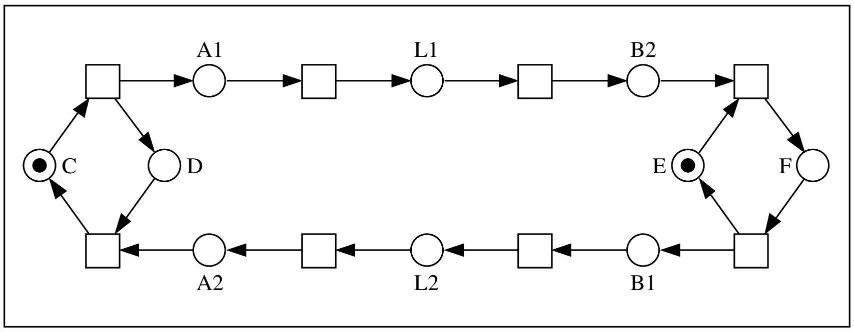

Figure 3: Composition of the above components

Hitherto, we have presented the syntactical aspects of components and their composition. The main aspect of this paper, however, is semantics: we want to formalize the behaviour (*semantics*) of a component in such a way, that the behaviour of a composed system can be derived from the behaviour of its components (without knowing the components themselves). Such a semantics is called *compositional* with respect to the composition of components.

The presentation of the compositional semantics needs some prerequisites, which will be given in the subsequent sections. In this informal example we will only motivate the main idea of this semantics: it is necessary to represent some behaviour of the environment of a component in the semantics of a component in order to make it compositional. To see this, we consider the component shown in Fig. 4, which has the same interface as the component *channel* from Fig. 1. When we consider the channel and the *empty* component in isolation (i.e. the environment doesn't put tokens to the input places), they both have the same behaviour: both components do absolutely nothing. Obviously, both components should show different behaviour, when composed with the producer and consumer component. So, if we want to derive the behaviour of a composed system only from the behaviour of its components, it is necessary that the channel and the empty component have different behaviour. Therefore, the semantics of a component *must* consider some aspects of the behaviour of the environment.

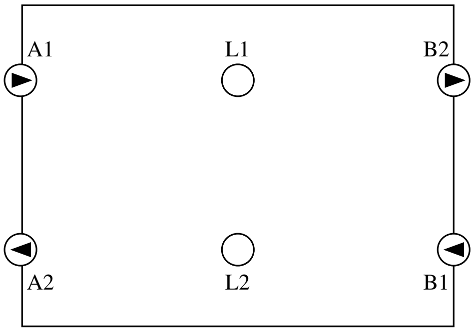

Figure 4: Empty component

In our approach we will adopt a very extreme position, where the semantics of a component explicitly represents the possible behaviour of its environment (cf. [BKP84, Pnu85, AL90]). Though, this approach seems to be naive at first glance, we will show that it is adequate from three different points of view:

Logical: Composition of two components corresponds to conjunction. This has very nice impacts on modular design methods [AL90, AL93] based on this semantics.

Semantical: Our semantics is *fully abstract* with respect to the composition operation and a *reference semantics* for closed components to be defined. Essentially, the reference semantics is the set of processes of a place/transition system [GR83] equipped with the *progress assumption* [Rei87, KW95, Rei95].

Practical: The semantics can be used to design systems in a modular way in combination with a temporal logic interpreted on processes of place/transition systems. In particular, we can specify components in a rely-guarantee-style.

2 Basic definitions

Now, we present the basic notions known from Petri net theory. We use place/transition systems (without capacities and arc-inscriptions) and their processes [GR83] as basic formalism. For simplicity, we consider only *T-restricted*[1] nets of *finite synchronization* [BF88]. Readers familiar with these concepts can quickly skim through this section.
We start with the definition of nets, markings and system nets.

Definition 1 (Net, marking, system net)

A triple $N = (P, T; F)$ is a *net*, if P and T are two disjoint sets and $F \subseteq (P \times T) \cup (T \times P)$.

A *marking* M of net N is a mapping $M : P \to \mathbb{N}$ such that $\sum_{p \in P} M(p) \in \mathbb{N}$.

A *system net* $\Sigma = (N, M)$ is a net N equipped with a marking M of N. The marking M is called the *initial marking* of Σ.

As usual, elements of P, T, and F are called *places*, *transitions* and *arcs* of the net, and are represented as circles, squares and arrows, respectively. A place or a transition of a net N is called *element* of N.

Definition 2 (Pre- and postset, minimal and maximal elements)

Let $N = (P, T; F)$ be a net and $x \in P \cup T$ be an element of N. The *preset* ${}^\bullet x$ and *postset* $x^\bullet$ of x are defined as ${}^\bullet x \mathrel{\hat{=}} \{y \in P \cup T \mid (y, x) \in F\}$ and $x^\bullet \mathrel{\hat{=}} \{y \in P \cup T \mid (x, y) \in F\}$. For a set of transitions $U \subseteq T$ we define ${}^\bullet U = \bigcup_{t \in U} {}^\bullet t$ and $U^\bullet = \bigcup_{t \in U} t^\bullet$.

The *minimal* and *maximal* elements of N are defined by ${}^\circ N \mathrel{\hat{=}} \{x \in P \cup T \mid {}^\bullet x = \emptyset\}$ and $N^\circ \mathrel{\hat{=}} \{x \in P \cup T \mid x^\bullet = \emptyset\}$, respectively.

A net N is *T-restricted*, if ${}^\circ N \cup N^\circ \subseteq P$. A net N is of *finite synchronization*, if for each transition $t \in T$ the preset and postset is finite.

For simplicity, we adopt the following restrictions:

Convention 1

We consider only nets $N = (P, T; F)$ which are T-restricted and of finite synchronization.

We fix a set S for the rest of this paper and consider only system nets $\Sigma = (N, M)$ with place set $P \subseteq S$ and $T \cap S = \emptyset$.

For a smooth presentation of processes of a system net, we introduce some notations for multisets.

---

[1] I.e. each transition of the net has a nonempty pre- and postset.

Definition 3 (Multiset notations)

A *multiset over* S is a mapping $M : S \to \mathbb{N}$. For $s \in S$ we write $M[s]$ instead of $M(s)$ to indicate that M is a multiset. We define the relation $\leq$ on multisets pointwise: $M \leq M'$, iff (if and only if) for each $s \in S$ holds $M[s] \leq M'[s]$. Moreover, we define the multiset $M + M'$ pointwise by $(M + M')[s] = M[s] + M'[s]$ for each $s \in S$. A multiset M is *finite*, if $\sum_{s \in S} M[s] \in \mathbb{N}$.

Since we assume $P \subseteq S$ for each net with places P, a marking M of a net can be considered as a multiset over S (with $M[s] = 0$ for $s \notin P$). Vice versa each finite multiset M over S with $M[s] = 0$ for $s \notin P$ can be considered as a marking of a net with places P.

Moreover, for a net $N = (P, T; F)$ and a transition $t \in T$ the preset ${}^\bullet t$ and postset $t^\bullet$ can be considered as multisets. To avoid confusion, we introduce new symbols to denote these multisets: ${}^-t : S \to \mathbb{N}$ and $t^+ : S \to \mathbb{N}$ with

$$ {}^-t[s] = \begin{cases} 1 & s \in {}^\bullet t \\ 0 & s \notin {}^\bullet t \end{cases} \qquad t^+[s] = \begin{cases} 1 & s \in t^\bullet \\ 0 & s \notin t^\bullet \end{cases} $$

for each $s \in S$.

Since we consider only T-restricted nets of finite synchronization, ${}^-t$ and t^+ are finite multisets and, therefore, are markings of the corresponding net.

A *process* of a system net is a labelled *occurrence net*, where the labelling establishes the correspondence between the occurrence net and the system net. First, we define an occurrence net.

Definition 4 (Occurrence net)

A net $K = (B, E; \lessdot)$ is an *occurrence net*[2], if

1. the transitive closure of $\lessdot$ is acyclic; the transitive closure of $\lessdot$ will be denoted by $<_K$,
2. for each $b \in B$ holds $|{}^\bullet b| \leq 1$ and $|b^\bullet| \leq 1$,
3. for each $b \in B$ the set $\{b' \in B \mid b' <_K b\}$ is finite, and
4. ${}^\circ K$ is finite.

For clarity, we call a place of an occurrence net *condition* and a transition of an occurrence net *event*.

---

[2] Remember that by Convention 1 we consider only T-restricted nets.

Definition 5 (Process of a system net)

Let $\Sigma = (N, M)$ be a system net with $N = (P, T; F)$, let $K = (B, E; <)$ be an occurrence net, and $r : B \to S$ be a *labelling (of conditions)*. For a finite subset $Q \subseteq B$ we denote by $r(Q)$ the multiset with $r(Q)[s] = |\{b \in Q \mid r(b) = s\}|$ for each $s \in S$.

The pair $\rho = (K, r)$ is a *process* of Σ iff

1. $r(^\circ K) = M$ and
2. for each $e \in E$ there exists a transition $t \in T$ such that $r(^\bullet e) = {}^-t$ and $r(e^\bullet) = t^+$.

3 Components and their semantics

In this section we formally introduce *components*, show how they are composed, and present a compositional semantics, which is based on the processes defined in the previous section.

3.1 Components

A component is a net with some addition: As shown in the example, we distinguish some places as *input* and *output* places of the component. Moreover, we distinguish two kinds of transitions: *progress* transitions and *quiescent* transitions. A progress transition [Rei95, KW95] of a component which is enabled forever must eventually occur. Most transitions of systems and components are assumed to be progress transitions. Therefore, we set off the quiescent transitions T^q in the formal definition.

Definition 6 (Component)

Let $\Sigma = (N, M)$ be a system net with $N = (P, T; F)$ and let $I, O \subseteq P$ be two disjoint sets with $M[p] = 0$ for each $p \in I \cup O$, and $^\bullet T \cap O = \emptyset$ and $T^\bullet \cap I = \emptyset$. Moreover, let $T^q \subseteq T$. We call $\Gamma = (\Sigma, T^q, I, O)$ a *component*. The component is *closed*, if $I = O = \emptyset$. A closed component is denoted by $\Gamma = (\Sigma, T^q)$, for short.

The elements of I are called *input places*, the elements of O are called *output places*. We have already seen the graphical representation of input and output places by an arrowhead into the corresponding direction. Note, that there is no conflict with the representation of the initial marking, because input and output places are initially unmarked. We call an element of $P \backslash (I \cup O)$ *internal* place and an element of $S \backslash P$ *external place* of Γ. A quiescent transition is represented by a dashed square — in contrast to progress transitions, which are represented as conventional squares.

As indicated in the example, two components can be composed when their internal places are disjoint and their common interface places have opposite directions. Then, the composed system is the union of both components, where common interface places become internal places of the composed system.

Definition 7 (Composition)

For $i \in \{1,2\}$ let $\Gamma_i = (\Sigma_i, T_i^q, I_i, O_i)$ be two components with $\Sigma_i = (N_i, M_i)$ and $N_i = (P_i, T_i; F_i)$.

The components Γ_1 and Γ_2 *have the same interface*, if $P_1 = P_2$, $I_1 = I_2$, and $O_1 = O_2$. The components Γ_1 and Γ_2 are *composable*, if $P_1 \cap P_2 = (I_1 \cap O_2) \cup (I_2 \cap O_1)$ and $T_1 \cap T_2 = \emptyset$.

For two composable components Γ_1 and Γ_2 the composed system $\Gamma = \Gamma_1 \Box \Gamma_2$ is defined by $\Gamma = (\Sigma, T^q, I, O)$ with $\Sigma = (N, M)$ and $N = (P_1 \cup P_2, T_1 \cup T_2; F_1 \cup F_2)$, and $M = M_1 + M_2$, $T^q = T_1^q \cup T_2^q$ and $I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1)$ and $O = (O_1 \setminus I_2) \cup (O_2 \setminus I_1)$

Note, that names of transitions are irrelevant in the semantics of a component, which will be defined later on. Therefore, we assume without loss of generality $T_1 \cap T_2 =$ for two components.
It can be easily checked, that the composition $\Gamma_1 \Box \Gamma_2$ of two composable components is a component again. Moreover, $\Box$ is commutative and associative.

3.2 Semantics

Before defining the compositional semantics for components, we define a *reference semantics* for closed components. This is the semantics in which we are actually interested. The compositional semantics, which we will present later on, is just a tool to reason about this reference semantics in a compositional way.
Essentially, the semantics of a closed component $\Gamma = (\Sigma, T^q)$ is the set of processes of Σ as defined in Sect. 2. The only additional requirement concerns the progress transitions of Γ: A progress condition must occur when possible (i.e. at the "end" K° of a run $\rho = (K, r)$ only quiescent transition are enabled). We call a process of Σ satisfying this additional requirement *closed run* of Γ.

Definition 8 (Reference semantics)

Let $\Gamma = (\Sigma, T^q)$ be a closed component with $\Sigma = (N, M)$ and $N = (P, T; F)$. A process $\rho = (K, r)$ of Σ is a *closed run* of Γ, if for each transition $t \in T$ and each finite set $Q \subseteq K^\circ$ with $r(Q) = {}^-t$ holds $t \in T^q$.

The set of all closed runs of Γ is denoted by $R_{cl}(\Gamma)$.

There are many different temporal logics, which are interpreted on runs. In some way each logic defines when a run ρ *satisfies* a temporal formula φ, which is denoted by $\rho \models \varphi$. We need to know only this relation in order to define the validity of a formula for a closed component: A formula φ is valid for a closed component Γ, if each run $\rho \in R_{cl}(\Gamma)$ satisfies φ. Therefore, the reference semantics is the formal basis for any of these temporal logics. At the end of this paper, we will give a concrete example for such a temporal logic.

Next, we define a compositional semantics for arbitrary components. As we have seen in the introduction, it is necessary to represent some behaviour of the environment in this semantics in order to achieve compositionality. Therefore, an event of a run of a component either corresponds to a transition of the component itself, or it corresponds to an event of its environment. This is captured by distinguishing two cases (2a. and 2b.) in the following definition.

Definition 9 (Semantics for components)

Let $\Gamma = (K, T^q, I, O)$ be a component with $\Sigma = (N, M)$ and $N = (P, T; F)$, let $K = (B, E; <)$ be an occurrence net, and let $r : B \to S$ be a labelling such that

1. $r({}^\circ K)[s] = M[s]$ for each $s \in P$.
2. For each event $e \in E$ one of the following conditions holds:
   (a) There exists a transition $t \in T$ such that $r({}^\bullet e) = {}^-t$ and $r(e^\bullet) = t^+$.
   (b) $r({}^\bullet e)[s] \leq 1$ and $r(e^\bullet)[s] \leq 1$ for each $s \in S$. Moreover, $r({}^\bullet e)[s] = 1$ implies $s \in O \cup (S \setminus P)$ and $r(e^\bullet)[s] = 1$ implies $s \in I \cup (S \setminus P)$.
3. For each transition $t \in T$ and each finite set $Q \subseteq K^\circ$ with $r(Q) = {}^-t$ holds $t \in T^q$.

Then, $\rho = (K, r)$ is a *run* of Γ. The set of all runs of Γ is denoted by $R(\Gamma)$.

Requirement 1 of Def. 9 corresponds to requirement 1 of Def. 5. The difference is that, here, we do not restrict the initial marking for external places (which is not known, because they are part of the environment). Requirement 2.a of Def. 9 corresponds to requirement 2 of Def. 5. However, the other case 2.b allows events that correspond to transitions of the environment. The last requirement concerning progress transitions is identical to the one of closed runs in Def. 8.

A main feature of this semantics is that composition of systems semantically means conjunction. This feature is crucial for modular reasoning in combination with some temporal logic, which will become more evident in Sect. 4.

Proposition 10 (Composition is conjunction)

Let Γ_1 and Γ_2 be two composable components. Then we have $R(\Gamma_1 \Box \Gamma_2) = R(\Gamma_1) \cap R(\Gamma_2)$.

Proof: Let Γ_1 and Γ_2 be composable components and $\Gamma = \Gamma_1 \Box \Gamma_2$ as in Def. 7 and let $\rho = (K, r)$. We have to show: $\rho \in R(\Gamma)$ iff $\rho \in R(\Gamma_1)$ and $\rho \in R(\Gamma_2)$. This can be done for each of the requirements 1–3 of Def. 9 separately.

1. Since Γ_1 and Γ_2 are composable components, we know $P_1 \cap P_2 \subseteq I_2 \cup O_2$. By definition of a component, we know $M_2[s] = 0$ for $s \in I_2 \cup O_2$. Thus by definition of the initial marking of the composed system, $M[s] = M_1[s]$ for each $s \in P_1$. Symmetrically, we get $M[s] = M_2[s]$ for each $s \in P_2$.

By definition of the composed system we know $P = P_1 \cup P_2$. Then, $r(^\circ Q)[s] = M[s]$ for each $s \in P$, iff $r(^\circ Q)[s] = M_1[s]$ for each $s \in P_1$ and $r(^\circ Q)[s] = M_2[s]$ for each $s \in P_2$.

2. First, we state some relations between sets. By the definition of composability, we know $P_1 \cap P_2 = (I_1 \cap O_2) \cup (I_2 \cap O_1)$. By elementary set theory we get $P_1 = (P_1 \setminus P_2) \ \cup \ (I_1 \cap O_2) \ \cup \ (I_2 \cap O_1)$. Then, $P_1 \setminus O_1 \subseteq S \setminus P_2 \ \cup \ O_2$ and $P_1 \setminus I_1 \subseteq S \setminus P_2 \ \cup \ I_2$.

   Now, let e be some event of K

   $\Rightarrow$ First, we assume that e satisfies requirement 2.a. for Γ. I.e. there exists a $t \in T$ with $r(^\bullet e) = {}^-t$ and $r(e^\bullet) = t^+$. W.l.o.g we assume $t \in T_1$; then, 2.a. is satisfied for e and component Γ_1.
   Now, we show that 2.b. holds for e and Γ_2. By $r(^\bullet e) = {}^-t$ and $r(e^\bullet) = t^+$ for $t \in T_1$, we know $r(^\bullet e)[s] \leq 1$ and $r(e^\bullet)[s] \leq 1$ for each $s \in S$. Moreover, since Γ_1 is a component with transition t, ${}^-t[s] = 1$ implies $s \in P_1 \setminus O_1 \subseteq S \setminus P_2 \cup O_2$ (by above properties) and $t^+[s] = 1$ implies $s \in P_1 \setminus I_\rho \subseteq S \setminus P_2 \cup I_2$.

   Now, we assume that e satisfies requirement 2.b. for Γ; since $s \in O \cup (S \setminus P)$ and $s \in I \cup (S \setminus P)$ immediately imply $s \in O_1 \cup (S \setminus P_1)$ and $s \in I_1 \cup (S \setminus P_1)$, respectively, it follows immediately, that e satisfies requirement 2.b. for Γ_1. Symmetrically, e satisfies requirement 2.b. for Γ_2.

   $\Leftarrow$ We assume that e satisfies requirement 2. for Γ_1 and Γ_2. If for one of them e satisfies requirement 2.a., then e obviously satisfies 2.a. for Γ (because $T = T_1 \cup T_2$).

   Now, we assume that e satisfies requirement 2.b. for, both, Γ_1 and Γ_2. Obviously, $r(^\bullet e)[s] \leq 1$ and $r(e^\bullet)[s] \leq 1$ for each $s \in S$.
   Assume $r(^\bullet e)[s] = 1$, then by 2.b. $s \in O_1 \cup S \setminus P_1$ and $s \in O_2 \cup S \setminus P_2$. Combining both properties (and $O_1 \cap O_2 = \emptyset$) gives us $s \in O_1 \cap (S \setminus P_2) \cup O_2 \cap (S \setminus P_1) \cup (S \setminus P_1) \cap (S \setminus P_2) = O_1 \setminus I_2 \cup O_2 \setminus I_2 \cup S \setminus (P_1 \cup P_2)$, which implies $s \in O \cup S \setminus P$. With similar arguments, we can prove $s \in I \cup S \setminus P$, if $r(e^\bullet)[s] = 1$.

3. By definition of components and their composition, we know $T = T_1 \cup T_2$, $T_1 \cap T_2 = \emptyset$, $T^q = T_1^q \cup T_2^q$, and $T_i^q \subseteq T_i$ for $i \in \{1, 2\}$

   $\Rightarrow$ We assume, that for each transition $t \in T$ and each finite set $Q \subseteq K^\circ$ with $r(Q) = {}^-t$ holds $t \in T^q$.
   Now, we additionally assume $t \in T_1$. Then, by $t \in T^q$ and the above properties, we conclude $t \in T_1^q$. Symmetrically, we get $t \in T_2^q$ if $t \in T_2$.

   $\Leftarrow$ Let $Q \subseteq K^\circ$ be a finite set. We assume that for $i \in \{1, 2\}$ and each transition $t \in T_i$ with $r(Q) = {}^-t$ holds $t \in T_i^q$. Then, by the above properties we conclude: $t \in T$ with $r(Q) = {}^-t$ implies $t \in T_q$. □

Another immediate consequence of Prop. 10 is the *compositionality* of the semantics for components, because the semantics of $\Gamma_1 \Box \Gamma_2$ can be derived (in a very easy way) from the semantics of its components. Therefore, our main effort to define a compositional semantics was successful.

Now, we are going to show that there is no better choice for a compositional semantics (with respect to our reference semantics). To this end, we show that the semantics R for components is *fully abstract* with respect to composition and the reference semantics R_{cl} for closed components.

Definition 11 (Fully abstract semantics)

A semantics S is *fully abstract* with respect to a composition operator $\Box$ and a semantics S', iff

1. S is *compositional*; i.e. there exists an operator $*$ such that for each two composable components Γ_1 and Γ_2 holds $S(\Gamma_1 \Box \Gamma_2) = S(\Gamma_1) * S(\Gamma_2)$.
2. S is *finer than* S'; i.e. for each two closed components Γ_1 and Γ_2 with the same set of places and $S(\Gamma_1) = S(\Gamma_2)$ holds $S'(\Gamma_1) = S'(\Gamma_2)$.
3. S is *as coarse as possible* with respect to requirements 1 and 2; i.e. for each two components Γ_1 and Γ_2 with the same interface and $S(\Gamma_1) \neq S(\Gamma_2)$ there exists a component Γ composable with Γ_1 and Γ_2 such that $\Gamma \Box \Gamma_1$ and $\Gamma \Box \Gamma_2$ is a closed component and $S'(\Gamma \Box \Gamma_1) \neq S'(\Gamma \Box \Gamma_2)$.

Theorem 12 (Fully abstract semantics)

The semantics R is fully abstract with respect to R_{cl} and $\Box$.

Proof:

1. Follows immediately from Prop. 10 with $* = \cap$.
2. For each closed component Γ with places P the set of runs $R_{cl}(\Gamma)$ is a restriction of $R(\Gamma)$ to runs $\rho = (K, r)$ with $r(b) \in P$ for each $b \in B$. Then, if $R(\Gamma_1) = R(\Gamma_2)$, the corresponding restrictions are equal, too.
3. If $R(\Gamma_1) \neq R(\Gamma_2)$, there exists a run $\rho = (K, \rho)$ which belongs to the one set and not to the other. W.l.o.g. we assume $\rho \in R(\Gamma_1)$ and $\rho \notin R(\Gamma_2)$. From this run ρ we construct a component Γ for which $\rho \in R_{cl}(\Gamma \Box \Gamma_1)$ and $\rho \notin R_{cl}(\Gamma \Box \Gamma_2)$, which proves $R_{cl}(\Gamma \Box \Gamma_1) \neq R_{cl}(\Gamma \Box \Gamma_2)$.

   The idea is the following: Γ contains all "external transitions" of ρ; i.e. transitions t with ${}^{-}t = r({}^{\bullet}e)$ and $t^{+} = r(e^{\bullet})$ for all events satisfying requirement 2.b. for Γ_1; we denote this set by T. We assume that all these transitions are quiescent; i.e. $T^q = T$. The initial marking M is $M[s] = r({}^{\circ}K)[s]$ for $s \in S \setminus P$ and $M[s] = 0$ for $s \in P$. The interface is "inverse" to that of Γ. So, $\Gamma = (\Sigma, T^q, O, I)$ with $\Sigma = (N, M)$ and $N = (S \setminus P \cup I \cup O, T; F)$.

Obviously, Γ is composable with Γ_1 and with Γ_2. It is easy to check, that ρ is a closed run of $\Gamma \Box \Gamma_1$. Moreover, ρ is no run of Γ_2, which implies $\rho \notin R(\Gamma \Box \Gamma_2)$ (Prop. 10). Since $R_{cl}(\Gamma \Box \Gamma_2) \subseteq R(\Gamma \Box \Gamma_2)$, we know $\rho \notin R_{cl}(\Gamma \Box \Gamma_2)$.

$\Box$

Theorem 12 proves the adequacy of the semantics R from a semantical point of view. We have already seen in Prop. 10, that composition corresponds to conjunction. This motivates the adequacy of our semantics from a logical point of view. Next we will show, that the semantics can be combined with a temporal logic for modular reasoning about distributed systems.

4 Towards a modular design formalism

Hitherto, we have presented components and their semantics. Next, we will show how to specify components by help of a temporal logic. This logic is similar to the Petri net logics proposed in [Rei95, GK+93, KW95]. But here, we consider runs which are not 1-safe, too. Moreover, we introduce a new operator for *partial invariants* [Kin95]. Partial invariants are used to establish invariants in a rely-guarantee-style, which will be shown later on.

4.1 Some more basic definitions

For the definition of the temporal logic we need some more basic definitions. A *state*[3] of an occurrence net is a set of conditions. The reachability of states from each other is defined as usual in condition/event-systems.

Definition 13 (States, reachable states)

Let $K = (B, E; <)$ be an occurrence net. A set $Q \subseteq B$ is a *state* of K.

For two *states* Q and Q' of K we say Q' is an *immediate successor* of Q, if there exists an event $e \in E$ such that ${}^\bullet e \subseteq Q$ and $e^\bullet \cap Q = \emptyset$ and $Q' = (Q \setminus {}^\bullet e) \cup e^\bullet$. Then, we write $Q \rightarrow Q'$. The relation $\overset{*}{\rightarrow}$ denotes the transitive and reflexive closure of $\rightarrow$. For two states Q and Q' of K with $Q \overset{*}{\rightarrow} Q'$ we say Q' *is reachable from* Q.

A state Q is *reachable in* K, if Q is reachable from ${}^\circ K$.

According to this definition each reachable state Q of an occurrence net is a finite set. Therefore, for a run $\rho = (K, r)$ and a reachable state Q of K the corresponding marking $r(Q)$ is well-defined. The operators of the temporal logic will be defined by help of the reachability relation on states and the markings corresponding to these states.

---

[3] We reserve the name "marking" for multisets, i.e. markings of place/transition systems. Therefore, we introduce a new name "state", here.

Before defining the temporal operators of the logic, we introduce a logic for markings. For a place $s \in S$ the symbol s is a term, which denotes a natural number; interpreted at a marking M the term s denotes the number of tokens $M[s]$ on place s. Moreover, each natural number $n \in \mathbb{N}$ is a term[4]. Two terms u and v can be combined to another term $u + v$.

From terms u and v we can build the proposition $u \leq v$. Moreover, we allow the usual boolean combinations of propositions. More formally we define:

Definition 14 (Terms and propositions)

1. The set of *terms* over S is denoted by $Term(S)$ and inductively defined by:
   (a) $S \subseteq Term(S)$.
   (b) $\mathbb{N} \subseteq Term(S)$.
   (c) If $u, v \in Term(S)$, then $u + v \in Term(S)$.
2. The set of *propositions* over S is denoted by $Prop(S)$ and inductively defined by:
   (a) If $u, v \in Term(S)$, then $u \leq v \in Prop(S)$.
   (b) If $\varphi \in Prop(S)$, then $\neg\varphi \in Prop(S)$.
   (c) If $\varphi, \psi \in Prop(S)$, then $(\varphi \wedge \psi) \in Prop(S)$.

As usual, the other boolean connectives can be considered as abbreviations. Similarly, $u = v$ and $u \geq v$ are abbreviations for $(u \leq v \wedge v \leq u)$ and $v \leq u$, respectively. For a finite set $X = \{x_1, \ldots, x_n\} \subseteq S$ we consider X as an abbreviation for the term $x_1 + \ldots + x_n$. The interpretation of terms and propositions at a marking is straightforward.

Definition 15 (Interpretation of terms and propositions)

1. For a marking $M : S \to \mathbb{N}$ we define a mapping $\overline{M} : Term(S) \to \mathbb{N}$, which associates each term with a natural number, inductively over the structure of the term:
   (a) For $s \in S$ we define $\overline{M}(s) = M[s]$.
   (b) For $n \in \mathbb{N}$ we define $\overline{M}(n) = n$.
   (c) For $u, v \in Term(S)$ we define $\overline{M}(u + v) = \overline{M}(u) + \overline{M}(v)$.
2. The validity of a proposition $\varphi \in Prop(S)$ at marking M is denoted by $M \models \varphi$ and defined inductively over the structure of the proposition:
   (a) For $u, v \in Term(S)$ we define $M \models u \leq v$, iff $\overline{M}(u) \leq \overline{M}(v)$.
   (b) For $\varphi \in Prop(S)$ we define $M \models \neg\varphi$, iff $M \not\models \varphi$.
   (c) For $\varphi, \psi \in Prop(S)$ we define $M \models (\varphi \wedge \psi)$, iff $M \models \varphi$ and $M \models \psi$.

---

[4] We don't fix a syntactical representation for natural numbers.

4.2 Temporal logic

Based on these propositions, we will define *temporal formulas*. For simplicity, we consider only unnested temporal formulas of the form φ, $\Box\varphi$ and $\varphi \leadsto \psi$ for propositions φ and ψ (cf. [KW95]). We call them *initial*, *invariant* and *leads-to* formulas, respectively.

In addition, we consider formulas of the form $\mathbf{p\text{-}i}(X, Y, Z)$, where $X, Y, Z \subseteq S$ are finite sets. We call those formulas *partial invariants*. The meaning of the above partial invariant is similar to that of a place invariant of a place/transition system. But we allow some places X (called *begin of the partial invariant*) at which tokens may enter the invariant and some places Z (called *end of the partial invariant*) at which tokens may leave the invariant. Two partial invariants with interchanged begin and end can be combined to a conventional place invariant. This way, partial invariants of different components can be combined to a place invariant of the composed system. This will be demonstrated by the following example.

Consider again the component *channel* from Fig. 1 in Sect. 1 and the composed system in Fig. 3. The composed system is closed and there is a place invariant which guarantees $C + A1 + L1 + B2 + F + B1 + L2 + A2 = 1$ for each reachable marking. The corresponding place invariant can be split into two parts, one part $\mathbf{p\text{-}i}(\{A1, B1\}, \{L1, L2\}, \{A2, B2\})$ concerning the component *channel* and the other part $\mathbf{p\text{-}i}(\{A2, B2\}, \{C, F\}, \{A1, B1\})$ guaranteed by its environment. The component *channel* only guarantees the first, but not the second partial invariant: it has no control over the external places C and F. For any environment of *channel* which guarantees the second partial invariant, we can combine both partial invariants, which gives us the above place invariant. Note, that $A1 + B1 + L1 + L2 + A2 + B2$ is a place invariant of *channel* in isolation. In an arbitrary environment, however, it is not a place invariant, because the environment may add tokens to $A1$ and $B1$ and may remove tokens from $A2$ and $B2$. Therefore, $A1$ and $B1$ are chosen as begin and $A2$ and $B2$ are chosen as end of the above partial invariant.

Definition 16 (Temporal properties)

The set of *temporal formulas* $TF(S)$ over S is defined by

1. Each proposition $\varphi \in \mathit{Prop}(S)$ is a temporal formula $\varphi \in TF(S)$.
2. If $\varphi \in \mathit{Prop}(S)$, then $\Box\varphi \in TF(S)$.
3. If $\varphi, \psi \in \mathit{Prop}(S)$, then $\varphi \leadsto \psi \in TF(S)$.
4. If $X, Y, Z \subseteq S$ are finite sets, then $\mathbf{p\text{-}i}(X, Y, Z) \in TF(S)$.

Let $\rho = (K, r)$ be a run of some component over S. Then we define the validity of a temporal formula φ at ρ, denoted by $\rho \models \varphi$, as follows:

1. For $\varphi \in \mathit{Prop}(S)$ we define $\rho \models \varphi$, iff $r({}^{\circ}K) \models \varphi$.
2. For $\varphi \in \mathit{Prop}(S)$ we define $\rho \models \Box\varphi$, iff for each reachable state Q of K holds $r(Q) \models \varphi$.

3. For $\varphi, \psi \in Prop(S)$ we define $\rho \models \varphi \leadsto \psi$, iff for each reachable state Q of K with $r(Q) \models \varphi$ there exists a state Q' reachable from Q such that $r(Q') \models \psi$.
4. For finite sets $X, Y, Z \subseteq S$ we define $\rho \models \mathbf{p\text{-}i}(X, Y, Z)$, iff for each event $e \in E$ of K holds
$$\sum_{s \in X \cup Y} r({}^\bullet e)[s] = \sum_{s \in Y \cup Z} r(e^\bullet)[s]$$

A component Γ satisfies a temporal formula φ, iff for each run $\rho \in R(\Gamma)$ holds $\rho \models \varphi$. Then,we write $\Gamma \models \varphi$.

From the definition of the validity of a partial invariant we can immediately proof the following simple[5] rule for combining partial invariants: If $\Gamma \models \mathbf{p\text{-}i}(X, Y_1, Z)$ and $\Gamma \models \mathbf{p\text{-}i}(Z, Y_2, X)$ for pairwise disjoint sets X, Y_1, Y_2, and Z, then $\Gamma \models \mathbf{p\text{-}i}(\emptyset, X \cup Y_1 \cup Y_2 \cup Z, \emptyset)$. Obviously, a partial invariant $\mathbf{p\text{-}i}(\emptyset, Y, \emptyset)$ corresponds to a place invariant, where each $y \in Y$ is weighted with 1.
Beside this rule, the logic enjoys two other properties that are crucial for a modular design of systems.

Theorem 17

1. Let Γ_1 and Γ_2 be two composable components and let $\varphi \in TF(S)$ be a temporal formula. If $\Gamma_1 \models \varphi$, then $\Gamma_1 \Box \Gamma_2 \models \varphi$.
2. Let Γ be a closed component with places P and $\varphi \in TF(P)$. Each *closed run* $\rho \in R_{cl}(\Gamma)$ of Γ satisfies φ, iff $\Gamma \models \varphi$.

The first property of Theorem 17 is an immediate consequence of Prop. 10. Note that it is independent of the choice of temporal operators and therefore holds for all temporal logics interpreted on runs. The second property of Theorem 17 slightly depends on the temporal operators of the logic. This property is important for the following reason: for uniformity, the validity of a formula for a closed component Γ was defined by help of $R(\Gamma)$ (as for all other components). Actually, we are only interested in the behaviour $R_{cl}(\Gamma)$ for closed components. Theorem 17 tells us, that both definitions coincide (when no external places occur in the temporal formula). Thus, the semantics in combination with this logic provides a method to argue about $R_{cl}(\Gamma)$ in a compositional way.

4.3 The Example: Revisited

Often, a component can only guarantee a property, when the environment behaves correctly. A specification of this kind is often called *rely-guarantee* specification [Jon83]. Here, we sketch a simple kind of rely-guarantee specification: for a component Γ, a temporal formula φ, and a set of temporal formulas Φ, we write $\Gamma, \Phi \models \varphi$, if component

---

[5] There is a more general rule [Kin95] for combining partial invariants with non-inverse begin and end, which is beyond the scope of this paper.

Γ guarantees φ, provided that the environment guarantees all properties of Φ. We will provide a flavour of this approach by an example.
We consider the example from the introduction again. We want to prove, that not both lines $L1$ and $L2$ of component *channel* are active (marked) at the same time. This property can be expressed by the temporal formula $\Box L1 + L2 \leq 1$. Of course this property doesn't hold for *channel* in an arbitrary environment. Therefore, we make an assumption about the environment: we assume, that there is a finite set X of external places such that the environment guarantees the partial invariant $\mathbf{p\text{-}i}(\{A2, B2\}, X, \{A1, B1\})$ and initially there is at most one token on the places of X, which is expressed by the temporal formula $X \leq 1$. These two assumptions are collected in $\Phi_1 = \{\mathbf{p\text{-}i}(\{A2, B2\}, X, \{A1, B1\}), X \leq 1\}$. Now, the component *channel* itself guarantees the partial invariant

$$\mathbf{p\text{-}i}(\{A1, B1\}, \{L1, L2\}, \{A2, B2\})$$

and from the initial marking of *channel* we know $A1 + L1 + A2 + B1 + L2 + B2 = 0$. When we combine both partial invariants we get the place invariant

$$\mathbf{p\text{-}i}(\emptyset, \{A1, L1, B2, B1, L2, A2\} \cup X, \emptyset)$$

In combination with the information about the initial state, we know $channel, \Phi_1 \models \Box A1 + L1 + B2 + B1 + L2 + A2 + X \leq 1$, which immediately implies $channel, \Phi_1 \models \Box L1 + L2 \leq 1$.
Similarly, we can prove $A1 \leadsto A2$ for *channel*, when we assume $B2 \leadsto B1$ from the environment ($\Phi_2 = \{B2 \leadsto B1\}$). To prove $channel, \Phi_2 \models A1 \leadsto A2$ we have to transitively combine the assumed property $B2 \leadsto B1$ with the properties $A1 \leadsto B2$ and $B1 \leadsto A2$ which are obviously true for *channel*.

5 Conclusion

The semantics presented in this paper was proposed in [Kin95]. There, the semantics was used as the semantical basis for modular design of distributed systems, where modules are specified in a rely-guarantee-style. Here, we could only sketch the application of the semantics. The emphasis of this paper is on the theoretical aspects of this semantics.
The basic idea of this semantics first appeared in combination with interleaving semantics [BKP84]. In interleaving semantics, however, there are some technical problems [AL90] (e.g. concerning fairness between transitions of the environment and transitions of the component, and a sophisticated labelling of events). These problems could be avoided in our approach by the use of partial order semantics in combination with the concept of components. Therefore, this semantics is more natural and doesn't need technical tricks.

The semantics $R(\Gamma)$ of a component Γ includes very many runs. But, we can easily handle these sets by help of a temporal logic — as illustrated in Sect. 4. In our logic we used partial invariants to prove invariants which are only guaranteed by two components in combination. It is necessary to have some kind of *one-step-operator* in the temporal logic to prove such invariants in a compositional way. Here, we have chosen partial invariants since they nicely fit to Petri nets. But, we could also use the more general *unless* operator for this purpose.

Partial invariants are similar to properties used in a client-server protocol [SB93] for the composition of Petri nets. But, the temporal logic (in combination with the compositional semantics) is more general than this client-server protocol. We feel, that the use of a design method based on our semantics are only limited by the power of the employed logic. In this sense, we have proposed a "general purpose" semantics for modular design formalisms.

Acknowledgments

I want to thank Rolf Walter and Hagen Völzer for many helpful suggestions. I believe their comments helped to improve the presentation of my ideas.

References

[AL90] Martín Abadi and Leslie Lamport. Composing specifications. In J.W. de Bakker and W.-P. de Roever, editors, *Stepwise Refinement of Distributed Systems. Models, Formalisms, Correctness*, *LNCS* 430. Springer-Verlag, 1990.

[AL93] Martín Abadi and Leslie Lamport. Conjoining specifications. Research Report 118, Digital Equipment Corporation, System Research Center, December 1993.

[BF88] Eike Best and César Fernández. *Nonsequential Processes*, *EATCS Monographs on Theoretical Computer Science* 13. Springer-Verlag, 1988.

[BKP84] Howard Barringer, Ruurd Kuiper, and Amir Pnueli. Now you may compose temporal logic specifications. In 16^{th} *annual ACM Symposium on Theory of Computing*, pages 51–63. ACM, April 1984.

[GK+93] Dominik Gomm, Ekkart Kindler, Barbara Paech, and Rolf Walter. Compositional liveness properties of EN-systems. In M. Ajmone Marsan, editor, *Applications and Theory of Petri Nets 1993*, *LNCS* 691, pages 262–281, June 1993. Springer-Verlag.

[GR83] U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.

[Jon83] Cliff. B Jones. Specification and design of (parallel) programs. In R.E.A Mason, editor, *Information Processing*, pages 321–332. IFIP, Elsevier Science Publishers B.V. (North Holland), 1983.

[Kin95] Ekkart Kindler. *Modularer Entwurf verteilter Systeme mit Petrinetzen*, *Edition Versal* 1. Bertz Verlag, December 1995. Dissertation, Technische Universität München.

[KW95] Ekkart Kindler and Rolf Walter. Message passing mutex. In J. Desel, editor, *Structures in Concurrency Theory*, Workshops in Computing, pages 205–219, Berlin, May 1995. Springer-Verlag.

[Maz88] Antoni Mazurkiewicz. Compositional semantics of pure place/transition systems. In G. Rozenberg, editor, *Advances in Petri Nets 1988*, *LNCS* 340, pages 307–330. Springer-Verlag, 1988.

[Pan90] Paritosh K. Pandya. Some comments on the assumption-commitment framework for compositional verification of distributed programs. In de Bakker, de Roever, and Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, *LNCS* 430, pages 622–640. Springer-Verlag, 1990.

[Pnu85] Amir Pnueli. In transition from global to modular temporal reasoning about programs. In K.R. Apt, editor, *Logics and Models of Concurrent Systems*, *Series F: Computer and System Science* 13, pages 123–144. Springer-Verlag, 1985.

[Rei87] Wolfgang Reisig. *Das Verhalten verteilter Systeme.* GMD-Bericht 170. Oldenbourg Verlag, 1987.

[Rei92] Wolfgang Reisig. Elements of a temporal logic coping with concurrency. SFB-Bericht 342/23/92 A, Technische Universiät München, November 1992.

[Rei95] Wolfgang Reisig. Petri net models of distributed algorithms. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, *LNCS* 1000, pages 441–454. Springer-Verlag, 1995.

[SB93] C. Sibertin-Blanc. A client-server protocol for the composition of Petri nets. In M-Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, *LNCS* 691, pages 377–396. Springer-Verlag, June 1993.