Composing Chameleons

Jörg Desel

FernUniversität in Hagen, Germany

Abstract

The most common composition mechanism for Petri nets synchronizes transitions of two nets by merging them: Two transitions are merged into one whose pre-set places consists of all pre-set places of the two transitions in their respective nets, and accordingly for the post-set places. This approach seems appropriate in applications where components are permanently connected, and therefore their models can be composed. However, it becomes very complex and even unsuitable when dealing with dynamically changing connection relations.

This short paper introduces a different and novel composition mechanism tailored for dynamic composition of Petri nets. We will call it *interaction*. The core idea is that when Petri net components interact, corresponding (interacting) transitions are merged, as is the case for composition, but the original transitions are preserved and available for further interactions.

Petri net components that are interpreted as parts of larger nets are called Petri net modules. We distinguish between internal and external transitions of Petri net modules, whereby only external transitions are capable of interaction. While the potential behavior of a module considers all transitions, the behavior of a closed Petri net, which does not provide for any further interaction, is limited to the internal transitions.

The approach is defined and illustrated using the examples of a vending machine and of sets of chameleons, which can only change their color cooperatively.

Keywords

Petri net composition, Petri net modules, interaction, chameleons

1. Introduction

An introductory lecture on Petri nets often begins with Petri net models of vending machines, see e.g. [1, 2]. An example (from the Petri Net Course at the Petri Net Conference) is shown in Figure 1. This example can explain a lot:

- There is a left hand component that describes the physical flow of goods and thus the physical part of a vending machine. This modeling is based on place/transition nets; the number of tokens in a place indicates how many goods are located at that position.
- The right hand component models the internal logic of the automaton, represented by an elementary net system. Places of this component are therefore to be interpreted as conditions that can either be true or false.
- The example shows that both Petri net components, although they have different Petri net types, can be merged at the common transition “dispense item”. The composed Petri net represents the overall behavior of the vending machine.
- The example was also created to represent alternatives (between “accept coin” and “reject coin”) and little concurrency (for example between “refill” and “accept coin”) in one figure.
- Considering the transitions “dispense item” separately in both components, both of them model an activity of the (hardware or software) system, that cannot occur without each other in the modeled composed system: No item can be dispensed without accepted money, and each signal “dispense item” should be accompanied by an actual dispense.
- There are further transitions of this Petri net that represent activities that also require an external partner. For example, one would not expect the system to insert a coin into itself. Therefore, one would need to merge this transition with a transition from a customer model. However, it also

PeNGE’25: Petri Net Games, Examples and Quizzes for Education, Contest and Fun. June 24, 2025, Paris, France
joerg.desel@fernuni-hagen.de (J. Desel)
https://www.fernuni-hagen.de/sttp/team/joerg.desel.shtml (J. Desel)

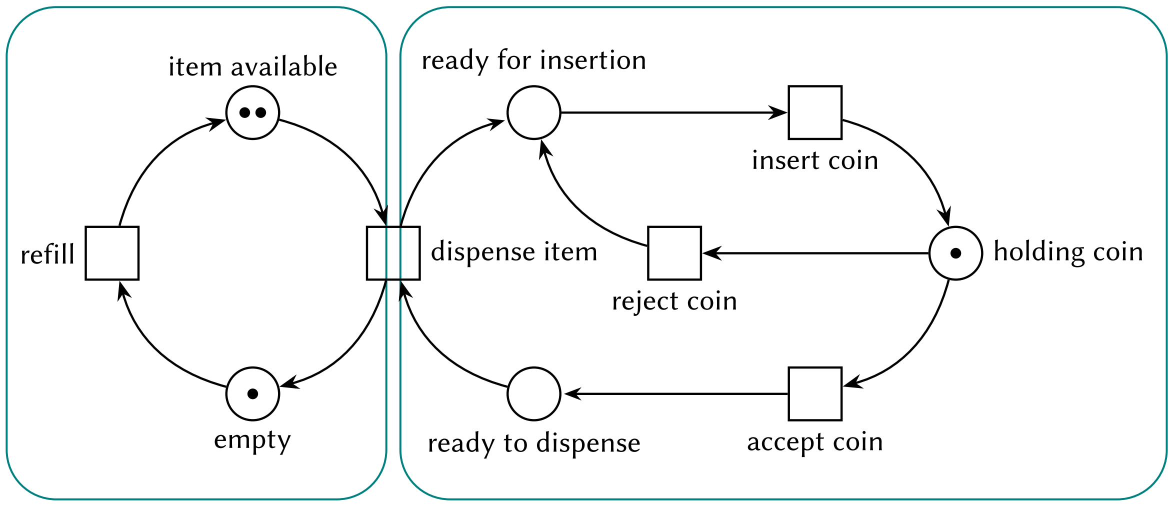

Figure 1: Petri net modul of a vending machine

makes sense to abstract from this interaction, as long as it is taken into account that the system cannot perform this activity on its own. The same applies to some other transitions, for example "refill". We will call such transitions *external*.

- For other activities, the responsibility lies solely with the modeled system. After a coin has been inserted, one can expect that the system proceeds without further interaction – the corresponding transitions "accept coin" and "reject coin" are said to be *internal*. Notice that the alternative between these two transitions is also considered internal. Although the criterion for coin acceptance is missing in the model and both transitions appear to occur randomly upon activation, the missing model of a coin acceptor is nevertheless internal.

Since nearly every modeled system has some relations to the outside world which is not in the focus of the model, a Petri net is usually just a model of a system component that is connected to potential other component models, but only this section of the overall model is shown. One could therefore assume that the transition "insert coin" is merged in a larger model with a transition of another Petri net – the model of a customer. However, this is not entirely true: While the two components of the vending machine model cooperate closely with each other, this is not the case when connecting to a model of a customer inserting a coin. A model composition would ensure that this would be the only customer forever – no other customer could dock to the "insert coin" transition, which is not the desired behavior. Instead, we need a dynamic composition where, in a single run (describing, for example, the behavior of a day), multiple customers use the machine, and thus multiple customer models can merge with the same transition of the model. The aim of this contribution is to develop and define such a compositional approach, which we will call *interaction*.

2. The Chameleon Game

As the title of this paper suggests, we will not use the vending machine as running example, but chameleons. The chameleon game, originally found in [3] and [4], was modeled and analysed with Petri nets in [5].

> Some chameleons live quite happily in a large terrarium. There are x red, y blue, and z green ones. These chameleons have a strange property: whenever two chameleons of different colors meet, both take on the third color. For example, when a blue and a green chameleon meet, they both turn red. And when a red and a green chameleon meet, they both turn blue.

In [5] termination and deadlock properties were studied, based on the respective initial numbers of red, blue and green chameleons. Now we want to model chameleons adequately to describe their behavior. Let us first provide a correct model for one chameleon:

As Figure 2 shows, a chameleon in a certain color can change to both other colors, resulting in a Petri net with three places (conditions) and six transitions. However, the rule of the chameleon game tells us that a chameleon cannot change color unless it contacts another chameleon. So, this model represents a single chameleon in some context, and all its transitions need to synchronize with transitions of other model components. We call all these transitions *external.* The Petri net is said to be a *Petri net module* (or *open Petri net*).

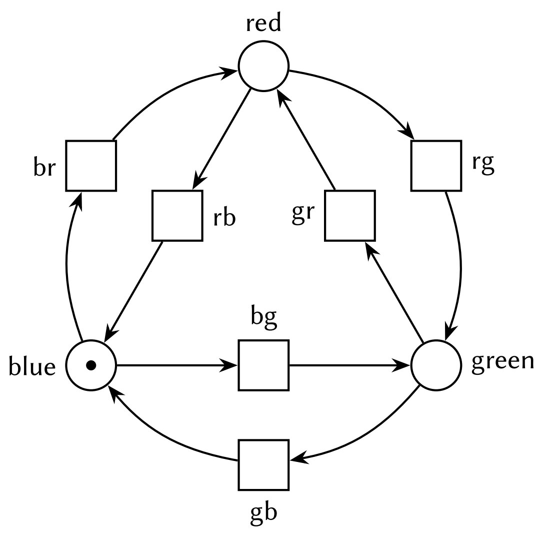

Figure 2: Petri net model of one chameleon

Now let us look at a model for two chameleons, built by composing two of the above modules. Since every color change of one chameleon (say $red \to blue$) corresponds to exactly one change of the other chameleon (here $green \to blue$), there is a one-to-one match between the respective sets of transitions. The resulting net is shown in Figure 3.

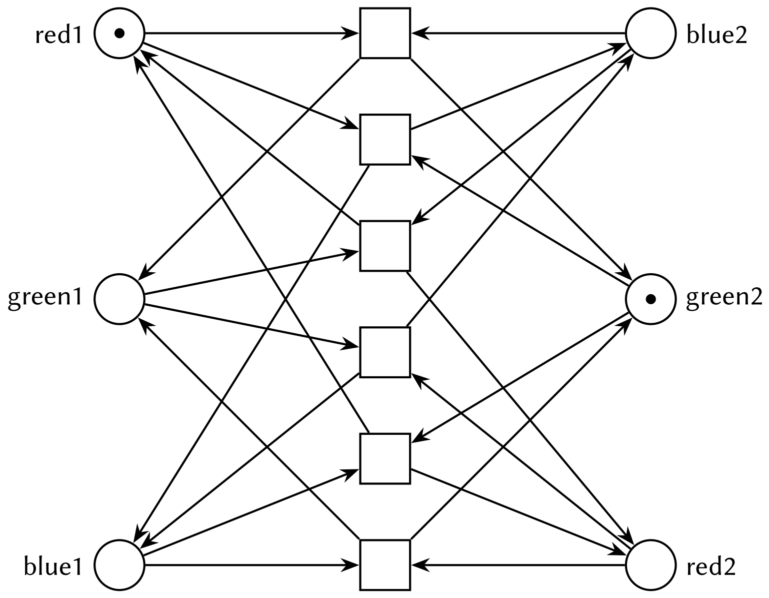

Figure 3: Petri net model of two chameleons

In this model, all transitions are *internal* because they cannot be composed with anything else. The net is a *closed Petri net*, which describes the complete behavior of two chameleons, provided that both will never contact a third one. We do not call this model a Petri net module because it cannot engage in any composition. However, its behavior is a bit boring: Either both chameleons have the same color and therefore no transition is enabled, or they have different colors (as depicted in the figure) and exactly one transition is enabled, the occurrence of which leads to a deadlock.

Next, we construct a model of two chameleons, which is extendable, i.e. we model two chameleons which can interact as the ones modeled in Figure 3, but can also encounter further chameleons. For this, the first composition of two chameleon models should not "consume" all transitions (making them internal), because this interaction may occur only once, and then the same transition should be able to synchronize with another transition. This consideration leads to a new composition operator, called *interaction* which is illustrated in the following figures:

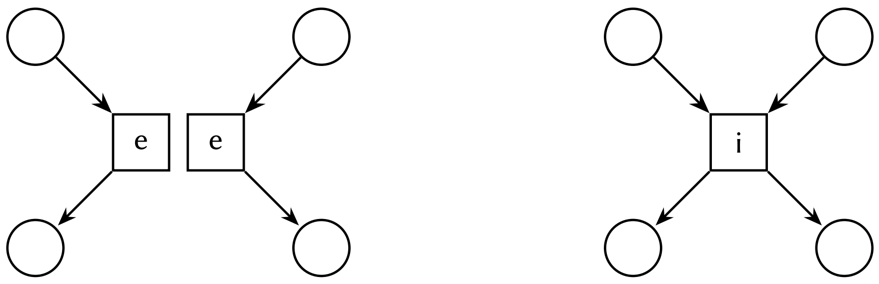

Figure 4: The traditional composition operator (e and i denote external and internal transitions)

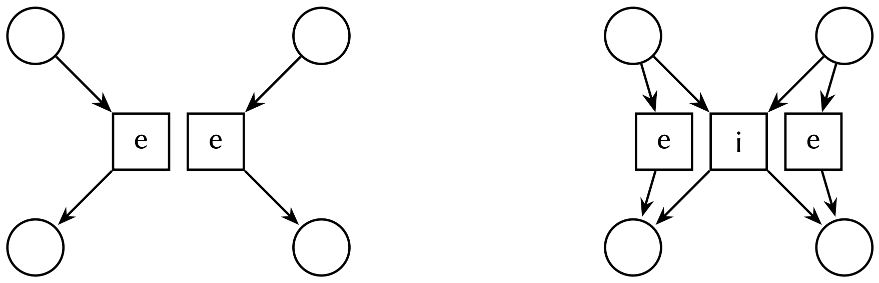

Figure 5: The interaction operator (e and i denote external and internal transitions)

If an external transition t_1 of a Petri net module N_1 interacts with an external transition t_2 of a Petri net module N_2, then the resulting Petri net will have all the net elements and arcs of the nets N_1 and N_2 (in particular, t_1 and t_2 are still external transitions), and additionally a new internal transition t satisfying ${}^\bullet t = {}^\bullet t_1 \cup {}^\bullet t_2$ and $t^\bullet = t_1^\bullet \cup t_2^\bullet$.

Merging two chameleon models by this interaction results in the Petri net shown in Figure 6. This is another open Petri net, i.e., a Petri net module with external transitions. The reader might argue that this Petri net interaction results in a Petri net with undesirable behavior: The internal transitions represent steps that are legal according to the rules of the chameleon game, whereas the external transitions represent illegal behavior, at least if the composed net is considered the final result. So we distinguish the potential behavior of a Petri net module given an arbitrary environment (*external behavior*) from the behavior of a Petri net which only uses its internal transitions (*internal behavior*). The internal behavior of the Petri net module shown in Figure 6 equals the behavior of the net of Figure 3. It represents the behavior of two chameleons which are the only ones, whereas the external behavior represents their potential behavior in a context.

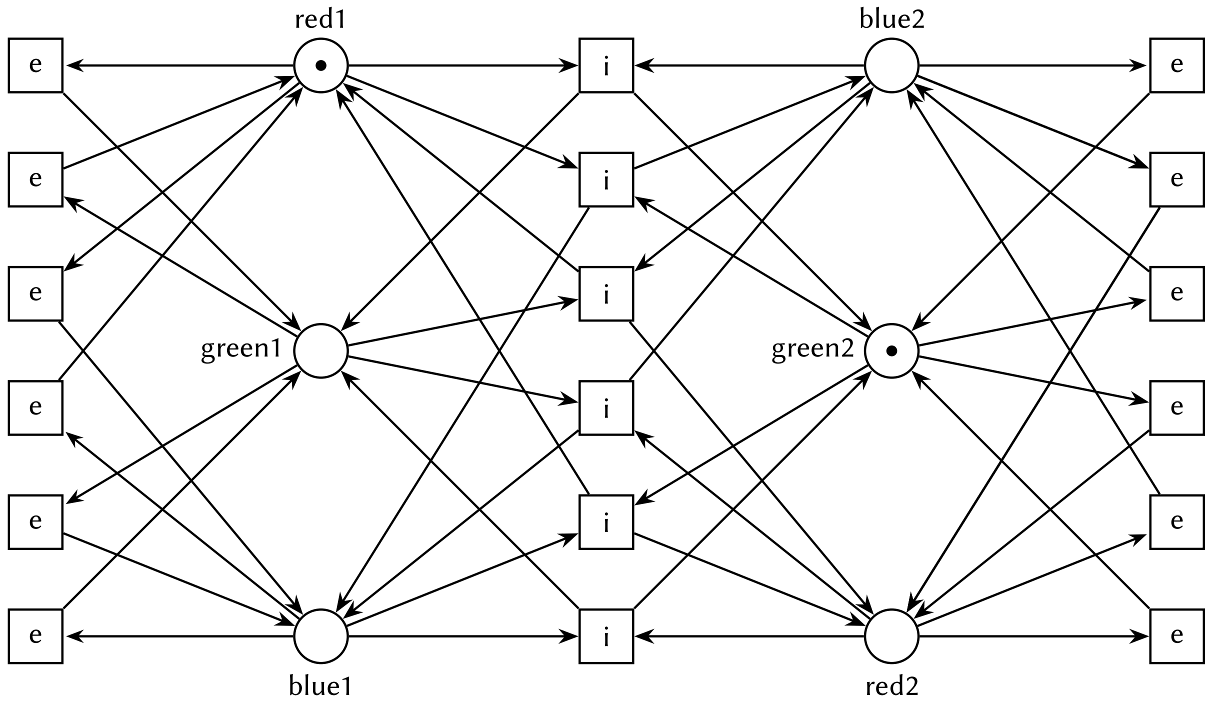

Figure 6: A Petri net module for two chameleons, external transitions are marked by "e"

Definition. A *Petri net module* is a Petri net $N = (S, T, F, M_0)$ with a set $T^e \subseteq T$ of *external* transitions. If $T^e \neq \emptyset$ then we call the module *open Petri net*, otherwise *closed Petri net*.

Assume two Petri net modules N_1 and N_2 with disjoint sets of elements and a relation $R \subseteq T_1^e \times T_2^e$. Their *interaction* $N_1 \circ N_2$ is defined by the unions of places, transitions, arcs, markings (viewed as relations), plus additional transitions (x, y) for each pair (x, y) in R. For each $(x, y) \in R$ and each arc (s, x) of N_1 or (s, y) of N_2 we add an arc $(s, (x, y))$, and for each arc (x, s) of N_1 or (y, s) of N_2 we add an arc $((x, y), s)$. The set of external transitions of $N_1 \circ N_2$ is the union of external transitions of the component nets, i.e., the new transitions are internal.

Lemma 1. *Petri net module composition is commutative, i.e.* $N_1 \circ_R N_2 = N_2 \circ_{R^{-1}} N_1$*, where* $\circ_R$ *denotes interaction with respect to relation* R*.*

Petri net module composition is associative, i.e. $(N_1 \circ_{R_1} N_2) \circ_{R_2} N_3 = N_1 \circ_{R_1} (N_2 \circ_{R_2} N_3)$*.*

These rules follow immediately from the definition of Petri net module interaction. Notice that these properties are not trivial when it comes to a syntactical representation of the relation R, as discussed in detail in the HERAKLIT approach [6].

By associativity, repeated use of the interaction operator allows to build arbitrarily large net modules. Since we assume disjoint nets in the definition, we must take care to add always "new" net modules, although these might be copies of the same pattern. This is the case for the chameleon example – each chameleon model needs new names for places and transitions. Instead of this assumption, we could have shifted disjointness into the construction: if nets are not disjoint, then the operator distinguishes the elements of the components before performing the actual composition. This allows to specify large nets syntactically by e.g. $N \circ N \circ N \circ N \circ N$ or even N^5.

Interaction of net modules allows to specify sets of modules that can interact arbitrarily, just in the sense of [7], but it is also possible to restrict to possible interactions. For example, we might be interested in modeling chameleons $c_0, c_1, \ldots, c_{n-1}$ such that chameleon c_i can only interact with chameleons $c_{i-1 (\mathrm{mod}\, n)}$ and $c_{i+1 (\mathrm{mod}\, n)}$, that is, chameleons arranged on a circle.

3. Behavior of interacting Petri net modules

What is the “natural” behavior of a Petri net module? For the vending machine, one would consider external transitions as first-class citizens of the game: the usual considerations include “insert coin” etc. In contrast, for the chameleons, one would not consider chameleons that change color without partner. So, it makes sense to consider both ways natural and to distinguish “internal behavior” (without external transitions) from “external behavior” (with external transitions).

Definition. Given a Petri net module, its *internal Petri net* is given by the Petri net of the module without external transitions (and incoming and outgoing arcs).

The *external behavior* of a Petri net module is the behavior of its Petri net.

The *internal behavior* of a Petri net module is the behavior of its internal Petri net.

We are interested in the external behavior of the vending machine model, but in the internal behavior of a model of the chameleon game. What about other examples? Consider a workflow Petri net model of a process as defined by van der Aalst [8], which has a unique (initially marked) initial place and a unique final place. Add an additional external transition that moves a token from the final place to the initial place. Then the internal behavior is the behavior of the workflow net itself, representing the behavior of the modeled process. Soundness is a behavioral property defined for internal behavior. External behavior also allows the occurrence of the “feedback transition”, which represents the environment of the process in a very abstract way. This external behavior makes it clear that a process should not be executed only once. Ideally, the Petri net (with the additional external transition) is live and bounded.

Typical behavioral properties such as liveness or deadlock-freeness refer to the definition of the behavior of a net. But do they refer to internal or to external behavior? We consider this question for deadlock-freeness in the sequel, but these considerations can be transferred to other properties like liveness as well.

A Petri net can run into a deadlock when it can reach a marking that does not enable any transition. A deadlock-free Petri net has no reachable deadlocks. To transfer this property to Petri net modules, we need to make two decisions, each with two possibilities: Should reachability refer to internal or external behavior, i.e. should we consider markings only reachable by occurrence sequences of internal transitions? And should the deadlock property (no transition enabled) refer to all transitions or to internal transitions only? So we have four options. We will show that all of them can be meaningful, depending on the system modeled.

- For the vending machine example, the state in which the machine is waiting for a customer would not be considered an undesired deadlock, even though the machine stops. For this example, reachability and deadlock refer to external behavior. Generally, this case refers to reactive models with interface.
- In the chameleon game, a deadlock occurs when all chameleons have the same color. Here both reachability and deadlock refer to internal behavior. Generally, this is the case when the system is fully modeled.
- For the model of a database system, one would include possible user interactions (external transitions) as part of behavior, but expect that the system never stops and waits for user input (deadlocked with respect to internal transitions). This case is typical for models where interaction is interruption.
- Finally, it might be desirable that a system cannot get stuck before the first user interaction. Then we consider reachability w.r.t. internal behavior and expect that for any reachable state either an internal or an external activity is possible. This property is relevant for the initialization phase of systems where interaction is possible but not mandatory.

4. Conclusion

We introduced the concept of interaction of Petri net modules and distinguished it from traditional composition. The distinction requires a distinction between internal and external transitions and between internal and external behavior. The examples illustrated that modeling with Petri nets does not simply involve drawing standalone Petri nets and observing their behavior, but that the modeled system, and thus also its model, is usually embedded in an environment. The interfaces to this environment play a crucial role in interpreting the model and defining its properties. A learner must therefore practice from the beginning to consider system interfaces and their role with regard to the model.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] J. Desel, J. Esparza, Free choice Petri nets, Cambridge University Press, USA, 1995.

[2] W. Reisig, Elements of distributed algorithms: modeling and analysis with Petri nets, Springer, 1998.

[3] H. Dambeck, M. Niestedt, Verwirrspiel im Terrarium, https://www.spiegel.de/karriere/verwirrspiel-im-terrarium-raetsel-der-woche-a-404e0560-ca83-418c-a5f3-8acfa3167d3d, 2021. Accessed: 2022-03-01.

[4] A. Beutelspacher, Das Geheimnis der zwölften Münze: Neue mathematische Knobeleien, C.H.Beck, 2021.

[5] J. Desel, The chameleon game, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2022, volume 3170 of *CEUR Workshop Proceedings*, 2022, pp. 202–210.

[6] P. Fettke, W. Reisig, Understanding the Digital World - Modeling with HERAKLIT, Springer, 2024.

[7] G. Berry, G. Boudol, The chemical abstract machine, Theoretical Computer Science 96 (1992) 217–248.

[8] W. M. P. van der Aalst, Verification of workflow nets, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, volume 1248 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 407–426.