Invariant Calculations for P/T-Nets with Synchronous Channels*

Simon Bott[1], Daniel Moldt[1], Laif-Oke Clasen[1] and Marcel Hansson[1]

[1]University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences, Department of Informatics
http://www.paose.de

Abstract

Partitioning of systems generally leads to simpler system models and reduced cost in subsequent computations. The control structure of such a system can be covered by P/T-nets. Enhancing them with synchronous channels allows for decoupling net modules via a kind of interface.

In this paper, we discuss how to extend former definitions of Place/Transition nets with synchronous channels and modular structure by general synchronization. Concerning modeling capabilities in terms of expressivity, the usage of synchronous channels introduces a more powerful modeling technique than traditional P/T-nets. Introducing new concepts requires discussing how traditional verification means are kept in the new formalism.

The main contributions of this paper are the extensions of the existing definitions of P/T nets with synchronous channels and inducing modular structures on the net templates. In addition, existing approaches of verification options regarding invariant computation are applied to our definitions of the model.

When modeling the control structures of a system using P/T nets with synchronous channels, the modules of the system can be covered with invariants by inducing a specific structure of the net modules. Restricting the possible interconnections of net modules can support verification with invariants.

Keywords

P/T-nets, Synchronous Channels, Structuring Mechanisms, Petri Net Modules, Modeling, Invariants, Reference Nets, Renew

1. Introduction

Petri nets and specifically P/T-nets (Place/Transition-nets) have proven themselves to be useful for modeling complex concurrent systems and to allow formal verification of properties. Invariants, as one well-known property, support the verification of various properties of such systems, for example, liveness and boundedness.

There are a variety of algorithms used for computing invariants that differ in their computational demands, resulting from different constraints desired for the resulting invariants. They range from methods based on Gaussian elimination [1] for computing any kind of basis of invariants to linear programming methods that compute minimal positive invariants [2, 3].

Another strategy for calculating invariants is the decomposition of systems, as in [4]. Here, a large system is decomposed into smaller subsystems and the corresponding calculations of invariants are carried out in the smaller subsystems.

The complexity resulting from modeling large systems makes the verification of such models computationally hard. It is the modular approach to modeling we want to consider in this paper. The division of large systems into smaller subsystems, or modules, reduces complexity by allowing individual modules to be examined independently of one another. However, the interactions between modules still have to be considered.

The model used in this paper is P/T nets with synchronous channels [5], whereby the definitions are refined within this paper. These are ordinary P/T nets that can communicate with each other via synchronous channels, similar to those of Christensen and Hansen [6, 7], which support synchronization of transitions.

PNSE'24: International Workshop on Petri Nets and Software Engineering, June 24–25, 2024, Geneva, Switzerland

*Supported by several colleagues and students.

The main question in this paper is how existing verification options and the modular approach of P/T nets with synchronous channels can be combined. The hypothesis is that decomposition based on the modular approach allows invariants within the subsystems to be calculated independently of each other, which means that these calculations can be distributed. The purpose of introducing such a concept covers more efficient verification in our tool and a structured / an object-oriented way of modeling by enhanced encapsulation via interfaces (modeled as synchronous channels.

In [8] and [9], some simple versions of invariant calculations have been introduced to Renew. The Ptc-nets (Place/Transition nets with Channels) are used and extended here in a simple manner. Additionally it is shown in [5] that Ptc-nets can be unfolded to P/T-nets and thus can model the same class of Petri net models as ordinary P/T-nets, however, in a more compact and structured way. In this paper, we present a variant of this and the modular nets of [10] and the Renew-based version in [5], we illustrate how P-invariants can be computed due to modularization, and we discuss the application of this modularized modeling to software modules. Our Petri nets-, Agent- and Organization-based Software Engineering approach (Paose, [11]) uses a special form of nets to model the communication of system components / agents. The Ptc-net variant introduced here can be used to support such system models.

The structure of the paper is the following: We first recall the definitions of P/T-nets and the equation from which invariants are derived in Section 3. Then we present our modular model in Section 4 informally with an example and with a formal definition, concluding in constructing an equivalent P/T-net. This leads to the examination of invariants in Ptc-system nets in Section 5, where we construct the incidence matrix by choosing a favorable ordering of its rows and columns and cover approaches to P- and T-invariants. In Section 6, we briefly show how invariants in Ptc-system nets can be used to verify some of their properties. Whereupon another example is explained in Section 7. In Section 8, we reflect on our findings and discuss further ideas concerning the application of the provided results. Finalized by the conclusion in Section 9.

2. Related Works

A common approach to limiting complexity is to identify and condense parts of nets that are similar in form and function. Higher-level nets like colored Petri nets serve such a purpose [12, 13]. colored Petri nets are extended upon to support synchronization through colored communication channels in [7]. Further, the formal definition of place invariants is extended to colored Petri nets with and without channels.

In this paper, the modular approaches to reducing the complexity of analysis are of particular interest.

In [10], modular P/T-nets are introduced, in which ordinary P/T-nets act as modules, and so-called place fusion sets and transition fusion sets define the possible interactions between modules. They allow the sharing of places and the synchronization of transitions, respectively. The resulting P-invariants and state spaces of modular P/T-nets are examined.

The state space construction of modular P/T-nets is expanded upon in [14] to also include shared places. The possibility of designing an algorithm that incorporates place fusions directly is investigated. However, such an algorithm would have similar results as those obtained by first transforming place fusions into transition fusions. Consequently, possible schemes for such transformations are analyzed.

In [15, 16], the modular approach is extended to colored Petri nets with the introduction of modular CP-nets. In comparison to modular P/T-nets, they contain colored Petri nets instead of P/T-nets as modules, with place fusion sets and transition fusion sets continuing to control interactions between modules. The resulting P-invariants and state spaces of modular CP-nets are examined.

In [5] P/T-nets with synchronous channels and modular PTC-nets are introduced. A formal definition and implementation in Renew are given for both models. The models are, however, limited in that only two transitions can be synchronized at a time. Further, verification is done by first constructing equivalent P/T-nets.

With HERAKLIT a new modular approached was introduced in [17] based on the composition

calculus and interface nets by Wolfgang Reisig [18]. In HERAKLIT each module is a net with two interfaces, which can contain places and transitions. A whole system is then composed by merging the same labeled elements of the interfaces of the modules. Merging two transitions this way is in some regards similar to connecting them via synchronous channels as the two transition are merged into one transition while with channels they behave as one transition. The looser coupling with channels allows a more dynamic behavior though as a transition could participate in different synchronizations.

Aside from Petri nets, modular approaches also exist for other models of describing concurrent processes. One example is the use of communicating sequential processes [19] in solving distributed constraint satisfaction problems [20].

Modeling complex systems strives to partition systems into smaller parts. The coupling of those parts should be loose compared to the internal coherent components. The Paose approach to model the control structure of systems via Petri nets with a clear interface supports this directly.

Mapping objects, agents, services, etc. to net modules is a straightforward approach. Modeling the communication between them via synchronous channels and P/T-nets allows for the reduction of the modeling to the orchestration and choreography of such system components. Encapsulation and abstraction of parts of the system by separating the channels is straightforward.

The Paose approach can incorporate this verification approach directly as our agents follow the design to communicate as specified by the corresponding Agent interaction protocol diagrams (Aips). Aips extend the UML sequence diagrams[21] by additional alternatives and concurrency. [11] Additionally, a formal semantics for Aips based on Petri nets was proposed in [22]. As discussed above these can be restricted to our Ptc-net definitions for verification purposes.

Currently, we are experimenting with the application to our Heraklit Agents [23]. These Heraklit Agents combine Reference Nets, Agents and HERAKLIT Modules [17].

3. Foundations and Problem Formulation

We start by defining P/T-nets and their invariants.

Definition 1. *A P/T-net is a tuple $N = (P, T, F, W, m_0)$, where:*

- P *is a finite set of places*
- T *is a finite set of transitions,* P *and* T *are disjoint:* $P \cap T = \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs*
- $W : F \to \mathbb{N}^+$ *is the arc weight function*
- $m_0 : P \to \mathbb{N}_0$ *is the initial marking*

It is helpful to extend the arc weight function in the following way:

Definition 2. *The extended arc weight function $\widetilde{W} : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ returns 0 for places/-transitions that share no directed arc:*

$$\widetilde{W}(x,y) = \begin{cases} W(x,y), & \text{if } (x,y) \in F \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Now we define the enabling and firing rules of a P/T-net.

Definition 3. *A transition t is enabled for a marking m, denoted by $m \xrightarrow{t}$, iff*

$$\forall p \in P : \widetilde{W}(p,t) \leq m(p) \tag{2}$$

Definition 4. *The successor marking $m'(p)$ after firing a transition t from a marking m, denoted by $m \xrightarrow{t} m'$, is defined as*

$$\forall p \in P : m'(p) = m(p) - \widetilde{W}(p,t) + \widetilde{W}(t,p) \tag{3}$$

At this point, we give another interchangeable definition of a P/T-net that consists of the backward incidence matrix $\mathbf{pre}$, describing the input of transitions, and the forward incidence matrix $\mathbf{post}$, describing the output of transitions[24]. It is going to be useful when constructing a P/T-net from our PTC-net.

Definition 5. *A P/T-net is a tuple $N = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$, where:*

- *P is a finite set of places*
- *T is a finite set of transitions, both sets are disjoint: $P \cap T = \emptyset$*
- *$\mathbf{pre}, \mathbf{post} \in \mathbb{N}_0^{|P|\times|T|}$ are the backward and forward incidence matrix respectively.*

The relation between this definition and Definition 1 is illustrated by expressing the elements of $\mathbf{pre}, \mathbf{post}$ through the extended arc weight function:

$$\mathbf{pre}_{pt} = \widetilde{W}(p, t) \tag{4}$$

$$\mathbf{post}_{pt} = \widetilde{W}(t, p) \tag{5}$$

The effect of a single transition is the net change that results from its firing. The effects of all transitions of a given net are aggregated in the incidence matrix. It is central to the study of a net's structural properties, such as invariants.

Definition 6. *The incidence matrix $\Delta \in \mathbb{Z}^{|P|\times|T|}$ of a P/T-net N is defined as:*

$$\Delta = \mathbf{post} - \mathbf{pre} \tag{6}$$

Now we can express the change of net markings in the form of an equation, where $\boldsymbol{x} \in \mathbb{N}_0^{|T|}$ contains the firing counts of each transition and m is the resulting marking.

$$m = m_0 + \Delta\boldsymbol{x} \tag{7}$$

T-invariants describe the special case when the net effect of a composition of firing counts is zero, or more specifically, the zero vector $\mathbf{0}$. P-invariants describe a weight function for the places of a net that is invariant with regard to the changes in markings resulting from firing transitions. Both can be expressed as solutions of a homogeneous system of equations:

$$\Delta\boldsymbol{x} = \mathbf{0} \tag{8}$$

$$\Delta^{\mathsf{T}}\boldsymbol{y} = \mathbf{0} \tag{9}$$

Where the non-trivial solutions $\boldsymbol{x} \in \mathbb{N}_0^{|T|} \setminus \{\mathbf{0}\}$ are T-invariants and the non-trivial solutions of $\boldsymbol{y} \in \mathbb{Z}^{|P|} \setminus \{\mathbf{0}\}$ are P-invariants. It is also common to only consider positive solutions for P-invariants.

We conclude with the definition of a multiset and the relevant notations used in this paper. Note that we shorten multisets to their functional components for brevity. A comprehensive overview of multisets can be found in [25].

Definition 7. *A multiset bg over a set D is a function $bg : D \to \mathbb{N}_0$.*

To differentiate a multiset from an ordinary set, we use a subscript b, for example, $\{a, b, b\}_b$.

Definition 8. *Let bg_1 and bg_2 be two multisets over the same set D, then their sum bg_3, denoted by $bg_3 = bg_1 \uplus bg_2$, is defined as:*

$$\forall d \in D : bg_3(d) = bg_1(d) + bg_2(d) \tag{10}$$

Definition 9. *Let D be a set, then we denote the set of all multisets over D as $Bag(D)$.*

Another type of net that will be referred to throughout the paper are reference nets which we will only describe with an informal description. Reference nets introduced in [26] are high-level Petri nets with arbitrary (Java) objects as tokens, that can be modeled and simulated with the tool Renew (The Reference Net Workshop) [27]. In particular, the tokens can be references to other net instances. The communication between net instances is done by synchronous channels (see [6, 7]).

Synchronous channels allow for synchronization of transitions. Transitions are equipped with channels and can communicate with other transitions with which they share a channel. Channels are primarily differentiated by their channel identifier. Additionally, channels are assigned a type, for example, *downlink* and *uplink*, so that transitions can only communicate with transitions with the same channel identifier of the opposite type. Lastly, parameters allow channels to specify a binding for potential variables that are weights of edges of communicating transitions [26].

4. Ptc-System Net

In the following we repeat the example and parts of the definition given in [5], where some different versions and extensions of [10] were given.

4.1. Informal Introduction

We now introduce Ptc-system nets using a simple producer/consumer example with a shared storage (Figure 1) constructed in Renew. Both producer and consumer have an internal logic that determines when they are able to produce or consume. For our example, we reduced them to the simple form where they are either ready or unready. The storage has a capacity and allows storing and retrieving as possible actions. The system net governs the interactions between modules.

Both the *Consumer* and *Producer* consist of a simple cycle. The transitions **tc0** and **tp0** are internal transitions and behave normally. The transitions **tc1** and **tp1**, however, are external, discernible from their inscriptions, which denote synchronous channels in Renew. They can't fire independently; instead, their behavior is controlled by the *SystemNet*.

The *Storage* consists of a place representing the content it is holding and its complementary place representing its available storage capacity. **ts0** allows storing things in the storage, while **ts1** allows retrieving things from the storage. Both are external transitions. Additionally, they contain variables that determine the amount transferred and are assigned indirectly by the *SystemNet*.

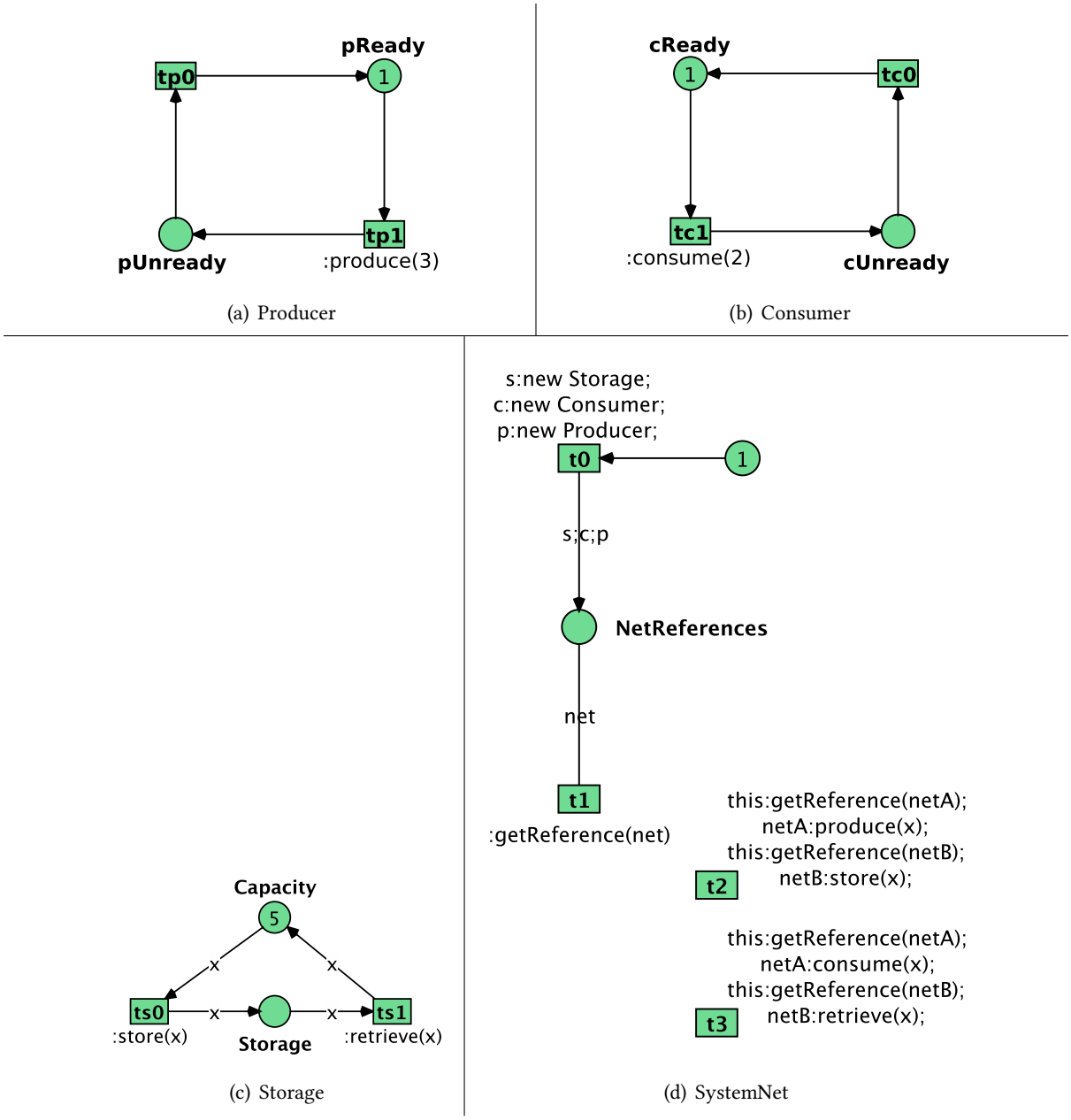
The *SystemNet* seems more complex than it really is due to its implementation in Renew. While the reference nets in Renew are more expressive than P/T-nets, we only use them in a restricted form. **t0** creates net instances of our modules and stores them in the place **NetReferences**. **t1** acts as an aid to address the individual modules. The key components of the *SystemNet* are the transitions **t2** and **t3**. They allow the synchronization between the external transitions of multiple modules. **t2** synchronizes **tp1** and **ts0** and binds the variable x to the value declared in the inscription of **tp1**. **t3** synchronizes **tc1** and **ts1** and binds the variable x to the value declared in the inscription of **tc1**. Both **t2** and **t3** represent synchronization rules governed by the Ptc-system net. (It should be pointed out that, while the binding of variables is done implicitly in Renew through parameters and unification, it is done explicitly in the proper formal definition of the Ptc-system net.)

We now go into the key differences in comparison to the *modular PT-net* presented in [10].

4.1.1. No shared places

To motivate the removal of shared places, we consider the concept of data encapsulation. It is desirable, for example, in object-oriented programming, to limit direct access to the fields/data of an object/module [28]. In P/T-nets these correspond to its passive elements, its places.

Furthermore, the synchronization of transitions alone is sufficient because they can emulate shared places. Instead of giving direct access to its places, a module can instead provide transitions that



(a) Producer (b) Consumer

(c) Storage (d) SystemNet

Figure 1: Producer and Consumer synchronization via a shared Storage; see [5]

conceptually act as getters/setters. In our example, the shared storage implements this idea. It provides places that are essentially shared, but does so through the sole use of synchronized transitions.

This transformation from shared places to synchronized transitions, however, is in general not obvious as shared places can be replaced in different ways. Such transformation schemes are studied in [14], with the best results in regard to modular state space construction being achieved when considering cohesion and coupling between modules.

Finally, we gain nice properties for P-invariants, already established in *Theorem 6.4* of [10] and further explored in Section 5.

4.1.2. Transitions participating in the same synchronization multiple times

Since the transition fusion sets in [10] are sets and not multisets, a given transition can only participate once. Allowing transitions to participate any number of times is a straightforward generalization.

This would be useful when an action of one module should be synchronized with the multiple

equivalent actions of other modules. If, for example, in our producer-consumer framework, a production process produces multiple goods at the same time, multiple store tasks, symbolized by synchronization with a store transition, would be necessary. From the view of the producer, it would be extraneous which modules in particular fulfill the store tasks. Especially whether it's synchronizing with multiple different storages or whether one single storage fulfills all store tasks.

4.1.3. Synchronized channels

The identification of similar behavior and its appropriate treatment is essential to limiting complexity. By assigning transitions that execute interchangeable actions the same channel, many different synchronizations can be established by a single synchronization rule. If, for example, we had two storages, the action of storing something provided by the individual storages would be considered interchangeable from an external view. The producer wouldn't care which storage responds to its communicated production. Therefore, if we wanted to add more modules representing producers or consumers to our example, we could do so without having to modify the synchronization rules of the PTC-system net. In addition, they naturally define an interface for each module.

It should be noted that while synchronized channels are used directly in the implementation in RENEW, they are hidden behind a layer of abstraction in the formal definition. This is done because the synchronized channels are used for the specific purpose of allowing communication between modules via the synchronization rules. Consequently, channels in modules are all of the same type, namely uplinks, and channels with differing names can be matched according to synchronization rules.

4.1.4. Variables as arc weights

Like the synchronized channels, the use of variables allows us to further aggregate actions of a similar function. If newly introduced producers/consumers differ by the amount they produce/consume, it suffices to introduce new possible bindings for the variables; no changes to the synchronization rules are needed. While the modular P/T-nets in [10] do not allow for such a functionality, the modular CP-nets in [15] do due to the use of colored Petri nets.

4.2. Formal Definition of PTC-System Nets

We now give the definition of a PTC-system net. But first, we define a P/T net module, which acts as the basic building block of the PTC-system net. Their difference from ordinary P/T-nets is that some transitions are external and can't fire independently. Instead, they are assigned channels to allow synchronization. Additionally, arcs involving external transitions are allowed to hold variables as weights.

Definition 10. *A P/T-net module is a tuple* $\mathcal{N} = (P, T, F, Var, W, m_0, K, T_e, E, f)$*, where:*

- P *is a finite set of places*
- T *is a finite set of transitions*
- $F \subseteq (P \times T) \cup (T \times P)$ *the flow relation*
- Var *is a finite set of variables*
- $W : F \to \mathbb{N}^+ \cup Var$ *the arc weight function*
- $m_0 : P \to \mathbb{N}_0$ *the initial marking*
- K *is a finite set of channels*
- $T_e \subseteq T$ *the set of external transitions*

- $E : T_e \to K$ *the channel assignment function*
- $f : T_e \to 2^{Var \times \mathbb{N}_0}$ *the variable assignment function*

Further, we require that transitions with variables on incoming or outgoing arcs are external transitions, or more formally:

$$\forall t \in T : (\exists v \in \{W(p,t) \mid p \in P\} \cup \{W(t,p) \mid p \in P\} : v \in Var) \Rightarrow t \in T_e \tag{11}$$

Now we have to adjust the enabling rules of a P/T-net module. Note that the extension $\widetilde{W}$ of the arc weight function remains unchanged.

Definition 11. *A transition t is enabled for a marking m, denoted by $m \xrightarrow{t}$, iff*

$$t \notin T_e \wedge \forall p \in P : \widetilde{W}(p,t) \leq m(p) \tag{12}$$

Now we get to the PTC-system net, the overarching model that includes all modules, the synchronization rules that govern interactions between modules and potential variable assignments for each rule.

Definition 12. *A PTC-system net is a tuple $\mathcal{SN} = (S, K, R, Var, f_R)$, where:*

- S *is a finite set of P/T-net modules, whose places and transitions are pairwise disjoint*
- K *is a finite set of channels*
- $R \subseteq Bag(K)$ *the synchronization rules*
- Var *is a finite set of variables*
- $f_R : R \to 2^{Var \times \mathbb{N}_0}$ *the variable assignment function*

Now follow some helpful definitions. First, some designators for the union of certain elements over all modules.

Definition 13. *For a given PTC-system net $\mathcal{SN} = (S, K, R, Var, f_R)$ with the P/T-net modules $s = (P_s, T_s, F_s, W_s, m_{0_s}, K_s, T_{e_s}, E_s, f_s) \in S$ the following helpful definitions are introduced:*

$$P = \bigcup_{s \in S} P_s \tag{13}$$

$$Var = \bigcup_{s \in S} Var_s \tag{14}$$

$$T = \bigcup_{s \in S} T_s \tag{15}$$

$$T_e = \bigcup_{s \in S} T_{e_s} \tag{16}$$

$$f : T_e \cup R \to Var \times \mathbb{N}_0 \tag{17}$$

$$f(x) = \begin{cases} f_s(x) & \text{, if } x \in T_{e_s} \\ f_R(x) & \text{, if } x \in R \end{cases} \tag{18}$$

Then we define the set of variables that is relevant when considering transitions that fire synchronously.

Definition 14. *The set of variables Var_{T_m} that are weights of incoming or outgoing arcs of a transition in a multiset of transitions T_m is defined as:*

$$Var_{T_m} = \{x \mid x \in Var \wedge \exists p \in P \exists t \in T_m : (x = W(p,t) \vee x = W(t,p))\} \tag{19}$$

When transitions synchronize, they are best described by a multiset of participating transitions, which we call a synchronized firing group. Further, it has to fulfill conditions in that it must fit a synchronization rule and a valid assignment for all relevant variables must exist. This is a key idea of the PTC-system net, because when we construct an ordinary P/T-net from a PTC-system net, every synchronized firing group is going to be its own transition.

Definition 15. *The set $G \subseteq Bag(T_e)$ contains the synchronized firing groups $g \in G$, satisfying:*

$$\exists r \in R : \left(\biguplus_{t \in g} \{E(t)\}_b \right) = r \tag{20}$$

$$\wedge\ \forall x \in Var_g : \exists! n \in \mathbb{N}_0 : (x, n) \in f(r) \cup \bigcup_{t \in g} f(t) \tag{21}$$

Equation 20 ensures that the participating transitions match a synchronization rule. Equation 21 describes how the assignments of relevant variables result from the individual assignments defined for each participating transition and the matching synchronization rule. Further, Equation 21 ensures that the assignment of each relevant variable exists and is unique.

In our example, Figure 1, the multiset $\{\mathbf{tp1}, \mathbf{ts0}\}_b$ is a valid synchronized firing group because the sum of their channels, $\{\text{produce}, \text{store}\}_b$ matches the synchronization rule, represented by **t2**, and the relevant variable x is uniquely assigned the value 3.

Definition 16. *The second condition of a synchronized firing group (Equation 21) is the definition of a function. Extension with the identity function for the natural numbers results in the function $\beta_{g,r} : Var_g \cup \mathbb{N}_0 \to \mathbb{N}_0$:*

$$\beta_{g,r}(x) = \begin{cases} x & , if\, x \in \mathbb{N}_0 \\ n & , if\, x \in Var_g \wedge (x, n) \in f(r) \cup \bigcup\limits_{t \in g} f(t) \end{cases} \tag{22}$$

This function $\beta_{g,r}$ gives an unambiguous binding for all relevant variables Var_g of any given synchronized firing group g. With it, we can now express the effect a firing group has on the marking of individual places.

Definition 17. *The arc weight function can be extended for the overarching PTC-system net to include the synchronized firing groups with the extension $\widetilde{W}_{sync} : (P \times G) \cup (G \times P) \to \mathbb{N}_0$:*

$$\widetilde{W}_{sync}(p, g) = \sum_{t \in g} \beta_{g,r}(\widetilde{W}(p, t)) \tag{23}$$

$$\widetilde{W}_{sync}(g, p) = \sum_{t \in g} \beta_{g,r}(\widetilde{W}(t, p)) \tag{24}$$

Now we conclude by defining the enabling and firing rules of a PTC-system net. Note that the enabling and firing rules of internal transitions,
$t \in T \setminus T_e$, follow from the definition of P/T-net modules (Definition 10).

Definition 18. *A synchronized firing group g is enabled for a marking m, denoted by $m \xrightarrow{g}$, iff*

$$\forall p \in P : \widetilde{W}_{sync}(p, g) \leq m(p) \tag{25}$$

Definition 19. *The successor marking $m'(p)$ after firing a synchronized firing group g, denoted by $m \xrightarrow{g} m'$, is defined as*

$$\forall p \in P : m'(p) = m(p) - \widetilde{W}_{sync}(p, g) + \widetilde{W}_{sync}(g, p) \tag{26}$$

We now shortly reexamine our example in Figure 1. The transitions **t2** and **t3** represent the two synchronization rules $r_{\mathbf{t2}}$ and $r_{\mathbf{t3}}$. These induce the synchronized firing groups $g_{\mathbf{t2}} = \{\mathbf{tp1}, \mathbf{ts0}\}$ and $g_{\mathbf{t3}} = \{\mathbf{tc1}, \mathbf{ts1}\}$ with the assignments $\beta_{g_{\mathbf{t2}}, r_{\mathbf{t2}}}(x) = 3$ and $\beta_{g_{\mathbf{t3}}, r_{\mathbf{t3}}}(x) = 2$.
$g_{\mathbf{t3}}$ is not enabled because $\widetilde{W}_{sync}(\mathbf{Storage}, g_{\mathbf{t3}}) = \beta_{g_{\mathbf{t3}}, r_{\mathbf{t3}}}(x) = 2 > m(\mathbf{Storage}) = 0$. $g_{\mathbf{t2}}$, however, is enabled, and the markings before and after firing $m \xrightarrow{g_{\mathbf{t2}}} m'$ are given in the following table:

p	pReady	pUnready	cReady	cUnready	Capacity	Storage
$m(p)$	1	0	1	0	5	0
$m'(p)$	0	1	1	0	2	3

4.3. Construction of an Equivalent P/T-Net

Our goal for this construction is the alternative definition of a P/T-net Definition 5. To that end, we construct **pre** and **post** for a given PTC-system net.

Definition 20. *The matrices* $\boldsymbol{pre}, \boldsymbol{post} \in \mathbb{Z}^{|P|\times(|T\setminus T_e|+|G|)}$ *for a given PTC-system net are as follows:*

$$\boldsymbol{pre}_{pt} = \widetilde{W}(p,t), \quad \text{with } t \in T \setminus T_e \tag{27}$$

$$\boldsymbol{pre}_{pg} = \widetilde{W}_{sync}(p,g) \tag{28}$$

$$\boldsymbol{post}_{pt} = \widetilde{W}(t,p), \quad \text{with } t \in T \setminus T_e \tag{29}$$

$$\boldsymbol{post}_{pg} = \widetilde{W}_{sync}(g,p) \tag{30}$$

The components of a P/T-net have to be finite. P and $T \setminus T_e$ are unions of finite sets and therefore finite themselves. That leaves only G.

Lemma 1. *The set of firing groups G for a given PTC-system net is finite, so $|G| < \infty$.*

Proof. Let $G' \subseteq Bag(T_e)$ be the set of firing groups $g' \in G'$, that would result if we were to ignore the fit of variables (equation 20). Their only condition is:

$$\exists r \in R : \left(\biguplus_{t \in g} \{E(t)\}_b \right) = r \tag{31}$$

If we also ignore the distinction between channels and consider the most complex synchronization rule $r_{max} \in R$ as the upper bound for all synchronization rules, that satisfies:

$$\forall r \in R : |r| \leq |r_{max}| \tag{32}$$

Then follows the upper limit:

$$|G| \leq |G'| \leq |R| \cdot |T_e|^{|r_{max}|} < \infty \tag{33}$$

□

Now we can define the equivalent P/T-net.

Definition 21. *For a given PTC-system net its equivalent P/T-net is given by the net* $\mathcal{N} = (P, (T \setminus T_e) \cup G, \boldsymbol{pre}, \boldsymbol{post}, m_0)$*, with* $m_0 : P \to \mathbb{N}_0$ *being the initial marking of all modules combined.*

5. Computing Invariants

Calculation of invariants is a costly operation. In [15, 16] and [8] some relevant explanations are given how modular nets could support the calculations. Here we describe the calculation for our slightly different definition.

5.1. Incidence Matrix

From Definition 20 we also get the incidence matrix $\Delta = \mathbf{post} - \mathbf{pre} \in \mathbb{Z}^{|P|\times(|T\setminus T_e|+|G|)}$. We now choose a suitable ordering for the elements of Δ by grouping places and internal transitions for each module and placing the synchronized firing groups at the end. This treatment was used already in [4].

We get the incidence matrix in the following form:

$$\Delta = \begin{pmatrix} \Delta_{s_1} & 0 & \cdots & \cdots & 0 & \Delta_{c_1} \\ 0 & \Delta_{s_2} & 0 & \cdots & 0 & \Delta_{c_2} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & \cdots & 0 & \Delta_{s_{n-1}} & 0 & \Delta_{c_{n-1}} \\ 0 & \cdots & \cdots & 0 & \Delta_{s_n} & \Delta_{c_n} \end{pmatrix} \tag{34}$$

The resulting matrix is now split into two parts. The first part is a block diagonal matrix with its submatrices Δ_{s_i} describing the internal actions of each module, while the second part is a block matrix with a single column containing submatrices Δ_{c_i} that describe the local effect of synchronized firing groups for each module.

Note that the resulting incidence matrix is in the desired form for the approach described in [4], in which a parallel algorithm for computing invariants is given.

5.2. Differing Approaches to P- and T-Invariants

The rows of submatrices of Δ can be viewed individually as the block matrices Δ_i and used to compute invariants for each module. This leads to the following homogeneous systems of equations:

$$\Delta_i = \begin{pmatrix} \Delta_{s_i} & \Delta_{c_i} \end{pmatrix} \tag{35}$$

$$\Delta_i \boldsymbol{x}_i = \mathbf{0}, \quad \text{with } \boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{x}_{s_i} \\ \boldsymbol{x}_{c_i} \end{pmatrix}, \boldsymbol{x}_{s_i} \in \mathbb{Z}^{|T\setminus T_e|}, \boldsymbol{x}_{c_i} \in \mathbb{Z}^{|G|} \tag{36}$$

$$\Delta_i^{\mathsf{T}} \boldsymbol{y}_i = \mathbf{0}, \quad \text{with } \boldsymbol{y}_i \in \mathbb{Z}^{|P_{s_i}|} \tag{37}$$

While the submatrix Δ_{c_i} might at first glance seem very large, the number of different non-zero columns is closer to $|T_{e_i}|$, the number of external transitions in the module s_i. Either synchronized firing groups don't include transitions of s_i, in which case the associated column is zero, or they include some linear combination of external transitions, in which case just the external transitions themselves need to be considered. Variables require a symbolic approach or add new non-zero columns for every different value they could be assigned. Therefore, the effort needed to compute the T-invariants for an individual module is comparable to the effort needed if it were an ordinary P/T-net.

We know from [4] that T-invariants $\boldsymbol{x}_i, \boldsymbol{x}_j$ of different modules s_i, s_j are the same part of a solution of the entire PTC-system net if and only if they have the same coupling components $\boldsymbol{x}_{c_i}, \boldsymbol{x}_{c_j}$. This allows us to combine local T-invariants to derive T-invariants for the whole system.

We know from *Theorem 6.4* in [10] that if $\boldsymbol{y}_i$ is a P-invariant of an individual module s_i then its extension for all places, where it is equal to zero for every place not in P_i, is also a P-invariant of the entire PTC-system net.

6. Verification

Because P-invariants for individual modules are preserved in the context of the whole PTC-system net, as shown in Section 5, properties that result from such P-invariants are preserved too.

One such property is the structural boundedness resulting from positive P-invariants. If we consider the *Producer* in our example Figure 1, we find that it has the P-invariant $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$. For the entire PTC-system

net, follows the P-invariant:

$$\begin{pmatrix}1\\1\\0\\0\\0\\0\end{pmatrix}, \quad \text{with the ordering} \quad \begin{matrix}\textbf{pReady}\\\textbf{pUnready}\\\textbf{cReady}\\\textbf{cUnready}\\\textbf{Capacity}\\\textbf{Storage}\end{matrix} \tag{38}$$

Since we have found a positive P-invariant spanning the places **pReady** and **pUnready**, we know that they are structurally bounded. The same can be done for *Producer* and *Storage*. Doing so results in the invariant that covers the whole net.

$$\begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix} \tag{39}$$

The internal logic of the modules in our example could also be expanded. We illustrate this in Figure 2, where we added a complex production process. The idea of no shared places as mentioned in 4.1.1 is also illustrated this way.

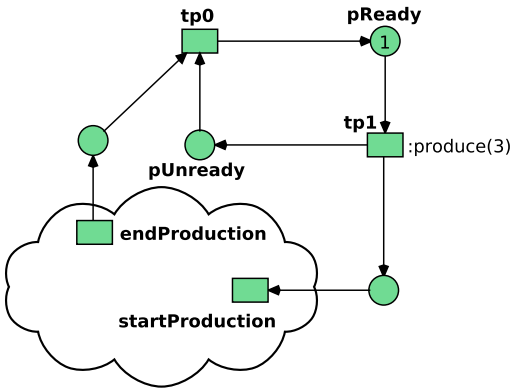


Figure 2: Producer with internal production

The place **pUnready** could be an implicit place that is refined by a whole subnet. The subnet contains the internal behavior of the net with the interface of transition **t1** and its uplink `:produce(3)`. All internal behavior is hidden through the transitions **startproduction** and **endproduction**. Further parts of the internal net are hidden. Assuming an invariant ensured in the internal net, this invariant can be calculated concurrently to all other internal nets in other net templates. It is important to notice that all P-invariants and T-invariants, provided they only include internal transitions, would be preserved in such internal processes.

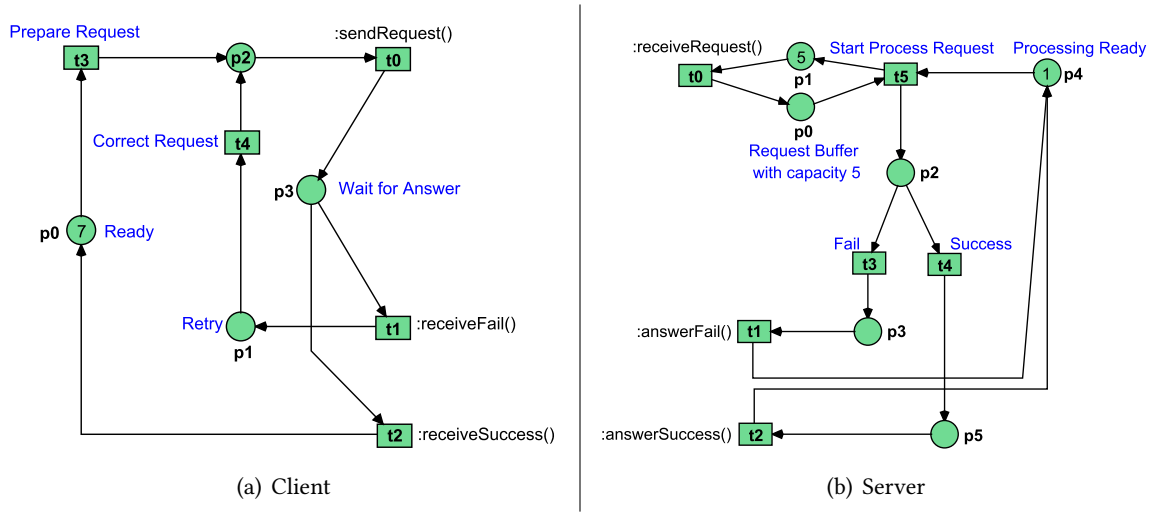
Furthermore, internal restriction on a net, e.g. being a correct workflow net (see [29]) could even speedup the calculations.

What has not been discussed so far and is also not covered by the presented solutions here is the dynamic creation of net instances of net templates. This is an inherent feature of reference nets [26] and our Renew tool. With slightly modified definitions this also becomes possible. However, the

mechanisms of introducing an arbitrary number of net instances provided by a net template as in reference nets would result in dynamic matrix dimensions. This is not covered by traditional invariant treatment and would make verification nearly impossible and hence we also do not cover this option here. In the following we will cover this feature partly when discussing the advantages for modeling.

7. Further Example

We will now consider one more example to further illustrate the PTC-system nets and verification method.



(a) Client (b) Server

Figure 3: A client sending request to the server and receiving success or fail responses.

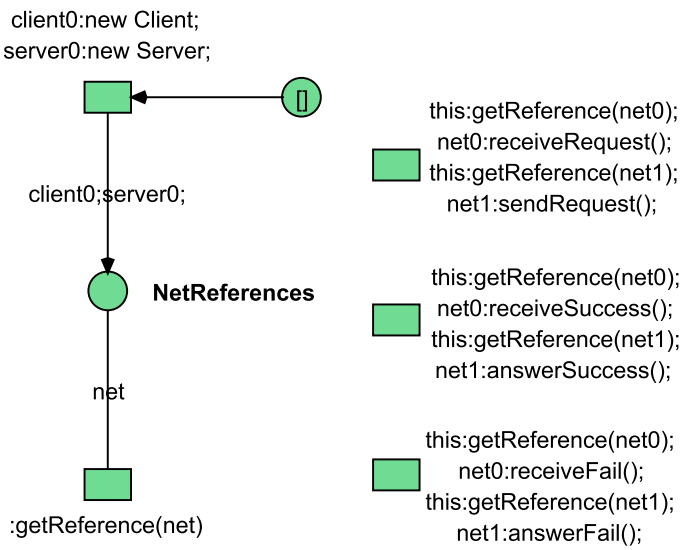


Figure 4: The system net creating the net instances.

The example shown in Figure 3 models a simple client-server interaction. The client sends requests (the details are abstracted away) to the server and either gets a fail or success as a response. The client can concurrently prepare and send seven requests due to the seven tokens on **p0**. After sending the request, a token is placed in **p3** to state that it is waiting for a response. If it receives a *fail* response, it corrects the request and sends it again; otherwise, it fills the *Ready* tokens on **p0** again. The request

exchange happens between the transitions that are named **t0** in both nets via the synchronous channels. The synchronization can be seen in the *SystemNet* shown in Figure 4.

On the server side, we have a request buffer (**p0**) to store up to five requests modeled with a complementary place (**p1**). The requests are processed one by one (increasing the tokens on **p4** would allow concurrent processing) and produce randomly either a fail or success response, which are modeled by different places. The responses are sent via the respective synchronous channels.

With the goal of finding invariants for the entire PTC-system net, we first consider the invariants of the individual modules. The elements of the invariants are ordered by the names of the transitions/places in ascending order.

For the Client net, we find two T-invariants $\boldsymbol{x}_{c,0}, \boldsymbol{x}_{c,1}$ and one P-invariant $\boldsymbol{y}_{c,0}$:

$$\boldsymbol{x}_{c,0} = \begin{pmatrix}1\\0\\1\\1\\0\end{pmatrix}, \boldsymbol{x}_{c,1} = \begin{pmatrix}1\\1\\0\\0\\1\end{pmatrix}, \qquad \boldsymbol{y}_{c,0} = \begin{pmatrix}1\\1\\1\\1\end{pmatrix} \tag{40}$$

For the Server net, we find two T-invariants $\boldsymbol{x}_{s,0}, \boldsymbol{x}_{s,1}$ and two P-invariants $\boldsymbol{y}_{s,0}, \boldsymbol{y}_{s,1}$:

$$\boldsymbol{x}_{s,0} = \begin{pmatrix}1\\0\\1\\0\\1\\1\end{pmatrix}, \boldsymbol{x}_{s,1} = \begin{pmatrix}1\\1\\0\\1\\0\\1\end{pmatrix}, \qquad \boldsymbol{y}_{s,0} = \begin{pmatrix}1\\1\\0\\0\\0\\0\end{pmatrix}, \boldsymbol{y}_{s,1} = \begin{pmatrix}0\\0\\1\\1\\1\\1\end{pmatrix} \tag{41}$$

We can now combine T-invariants with the same coupling components. They correspond to the transitions being synchronized, so $\mathbf{t0}_c$ and $\mathbf{t0}_s$, $\mathbf{t1}_c$ and $\mathbf{t1}_s$, $\mathbf{t2}_c$ and $\mathbf{t2}_s$ (we use subscripts to differentiate between modules). For example, $\boldsymbol{x}_{c,0}$ and $\boldsymbol{x}_{s,1}$ can't be combined because their values differ between $\mathbf{t1}_c$ and $\mathbf{t1}_s$. Their coupling components do not match. The coupling components of $\boldsymbol{x}_{c,0}$ and $\boldsymbol{x}_{s,0}$, however, do match, and we can combine them to form the new T-invariant $\boldsymbol{x}_0$. Similarly, we get $\boldsymbol{x}_1$ from $\boldsymbol{x}_{c,1}$ and $\boldsymbol{x}_{s,1}$.

The P-invariants stay preserved and can be expressed for the entire PTC-system net by extending them with 0-entries. For the order of elements, we list the places of the Client module in ascending order, followed by the places of the Server module in ascending order. Thereby, we get $\boldsymbol{y}_0$ from $\boldsymbol{y}_{c,0}$, $\boldsymbol{y}_1$ from $\boldsymbol{y}_{s,0}$, and $\boldsymbol{y}_2$ from $\boldsymbol{y}_{s,1}$.

$$\boldsymbol{x}_0 = \begin{pmatrix}1\\0\\1\\1\\0\\1\\0\\1\end{pmatrix}, \boldsymbol{x}_1 = \begin{pmatrix}1\\1\\0\\0\\1\\1\\1\\0\end{pmatrix}, \quad \text{with the ordering} \begin{matrix}\{\mathbf{t0}_c, \mathbf{t0}_s\}_b\\ \{\mathbf{t1}_c, \mathbf{t1}_s\}_b\\ \{\mathbf{t2}_c, \mathbf{t2}_s\}_b\\ \mathbf{t3}_c\\ \mathbf{t4}_c\\ \mathbf{t3}_s\\ \mathbf{t4}_s\\ \mathbf{t5}_s\end{matrix}, \qquad \boldsymbol{y}_0 = \begin{pmatrix}1\\1\\1\\1\\0\\0\\0\\0\\0\end{pmatrix}, \boldsymbol{y}_1 = \begin{pmatrix}0\\0\\0\\0\\1\\1\\0\\0\\0\end{pmatrix}, \boldsymbol{y}_2 = \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\1\\1\end{pmatrix} \tag{42}$$

8. Discussion

First of all it is important to notice that we do not work with traditional P/T-nets only. However, all presented versions can directly be mapped to P/T-nets. What is an important modeling advantage is that

the formalism is relatively simple compared to Colored Petri Nets (see [30, 31]) or reference nets (see [26]). Synchronous channels provide additional power in terms of modeling capabilities, introducing new modeling options to P/T-nets.

The approaches in [5, 9, 8] limited synchronization by allowing only two transitions to participate. This was done to avoid cyclic calls and possibly resulting infinite synchronizations. The PTC-system net also avoids such synchronizations, but without any such restriction.

As already identified in [10], the efficacy of the modular approach is inversely related to the extent of the synchronization between modules. In the worst case, every transition is external and has a unique channel. Then the PTC-system net would just describe a majorly interconnected P/T-net and nothing would be gained, as there would be no modular structure to exploit. In the best case, there is no interaction between modules at all. The PTC-system net would describe the union of its modules, and all modules could be viewed completely independently of one another. Well-structured nets will make limited use of the synchronization.

One interpretation of the synchronization is to consider the synchronization of two nets via the synchronous channel as the communication of two objects or agents. This idea has been shown in [32, 33, 34] and heavily used in subsequent work in this field (see [35]). However, there the focus was on expressibility when modeling complex systems with elaborated communication concepts related to speech act theory [36]. For the MULAN nets [37, 11] so called Agent Interaction Protocol Diagrams are used to describe the communication (what is very similar to sequence diagrams from UML [38]). In [23] the relation to the HERAKLIT approach is introduced for a version concentrating on the underlying P/T-net structure of the relations between agents / HERAKLIT modules. Therefore our presented results can be applied to this area of work. Abstraction of modules is easy. Behind a net module more complex system parts can be hidden. By separating the channels of external and internal parts, behind the interfaces of a net module, arbitrary large further sets of net modules can communicate completely separated from the rest of the system (if the channels are disjunct from the channels of net modules outside the considered net module).

Due to the high flexibility of the combination of synchronous channels, it is important to minimize the complexity of the synchronization. We believe this to be an advantage of the PTC-system net in comparison to [10]. Allowing n transitions with one channel α to synchronize with m transitions with another channel β would require only a single synchronization rule. Meanwhile, it would require a transition fusion set for every possible combination, $n \cdot m$. If the m transitions are all part of the same module and only differed by the values on some incoming and outgoing arcs, then our model could simplify them to a single transition that uses variables. Such a simplification is not possible in [10].

While we covered the preservation of P-invariants, there are also P-invariants that are newly introduced when introducing interaction between modules. These aren't as easy to compute because more than individual modules have to be considered. However, the modular structure is still beneficial in their computation. Firstly, the incidence matrices of modules can be reduced into a more favorable form independently of one another, enabling parallel computation. Secondly, the synchronous channels make such reductions easier by reducing the number of differing columns. This also covers the ideas presented in [5].

A possible goal would be the reduction to an upper triangular form. Such a form could be used in state space construction [39]. While a subset of generating P-invariants could also be used, the full set would further minimize the space required to store markings of the state space.

An alternative to conventional state space constructions would be the modular state space in [16, 10]. Because only modular P/T-nets without place fusions were considered, it is reasonable to expect that modular state spaces can also be applied to PTC-system nets. However, further examination is required to determine how this would work in detail and whether synchronous channels would bring significant improvements. This directly follows the ideas presented in [5].

The independence of modules in regard to their properties resulting from P-invariants could allow making statements about modules added during runtime. This would lead to a natural extension of the PTC-system net, where modules can be dynamically created and deleted with P/T-nets acting as patterns. To extend the net structure at execution / simulation time is an inherent feature of reference nets [26].

In Renew the idea of net templates that can be instantiated and deleted during the dynamic phase of a net, is an essential part of the tool [40]. The newly provided formalism for Renew covers the version of [5]. For our extension we are currently building an internal prototype to cover the formalism and to extend the work of [8]. Generally it is quite obvious that tool support with respect to modeling and verification are both very helpful. Another application, as also mentioned in [10, 5], is the independent calculation of the state space. However, in this contribution we do not deepen this discussion, as it falls in the same category as the abstraction of subsystems by separating the possibly shared channels.

Currently, we work on the application of our Ptc-net formalism to model the control flow between software modules to cover orchestration and choreography of participating software modules. Each software module gets a net module assigned. The calls of methods that are part of the interface of the software module are covered by synchronized transitions. The software modules must then be rigorously restricted in order to follow the behavior of the net modules. The net modules must cover the complex software behavior without handling the actual values and functions to allow the mapping between software and net modules. First prototypes of this are build and successfully tested and will be covered by other work of the authors and their colleagues.

9. Conclusion

An extension to former work of [10, 5] has been made to allow more complex synchronization of net modules that are coupled via synchronous channels. Due to the special perspective and used concepts the calculation of invariants are conceptually supported with respect to concurrent calculation of individual internal invariants and for the synchronization parts of the whole system. Our modular approach can allow for reduced complexity in the analysis of large systems.

Removal of shared places allows conserving P-invariants of modules, and the introduction of synchronous channels can allow similar actions in the model to be condensed. Multisets are used at various points to express our definitions and are especially useful in allowing transitions to participate in firing groups more than just once. Further, T-invariants for the entire net result from combining the T-invariants of individual modules.

Overall, verification can be done in a way that considers individual modules for most computations, thereby reducing runtimes and modeling of complex systems is directly supported by relating software modules / software components / services etc. to net modules.

Options to combine Petri net modeling (supported by verification) with software engineering to ensure properties of e.g. plugin architectures is ongoing work. As tool support is a necessary condition we also improve our current software infrastructure Renew and the whole modeling approach Paose concurrently to the more theoretical work.

References

[1] S. Cayir, M. Uçer, An algorithm to compute a basis of Petri net invariants, in: 4th ELECO Int. Conf. on Electrical and Electronics Engineering. UCTEA, Bursa, Turkey, 2005.

[2] J. Martínez, M. Silva, A simple and fast algorithm to obtain all invariants of a generalised Petri net, in: Application and Theory of Petri Nets: Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets Strasbourg, 23.–26. September 1980 Bad Honnef, 28.–30. September 1981, Springer, 1982, pp. 301–310.

[3] C.-F. Law, B. Gwee, J. S. Chang, Optimized algorithm for computing invariants of ordinary Petri nets, in: 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06), IEEE, 2006, pp. 23–28.

[4] A. Bourjij, M. Boutayeb, T. Cecchin, A decentralized approach for computing invariants in large scale and interconnected Petri nets, in: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, volume 2, IEEE, 1997, pp. 1741–1746.

[5] L. Voß, S. Willrodt, D. Moldt, M. Haustermann, Between expressiveness and verifiability: P/T-nets with synchronous channels and modular structure, in: M. Köhler-Bußmeier, D. Moldt, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2022 co-located with the 43rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2022), Bergen, Norway, June 20th, 2022, volume 3170 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 40–59. URL: https://ceur-ws.org/Vol-3170.

[6] E. Jessen, R. Valk, Rechensysteme: Grundlagen der Modellbildung, Studienreihe Informatik, Springer-Verlag, Berlin Heidelberg New York, 1987.

[7] S. Christensen, N. D. Hansen, Coloured Petri Nets Extended with Channels for Synchronous Communication, Technical Report DAIMI PB-390, Aarhus University, 1992.

[8] C. Künemund, Entwicklung eines Plugins zur Berechnung und Visualisierung von Invarianten in P/T-Netzen mit synchronen Kanälen in Renew, Bachelor thesis, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2021.

[9] L. Voß, Development of a Formalism for P/T Nets with Synchronous Channels and their Analysis using Siphons and Traps in Renew, Bachelor thesis, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2022.

[10] S. Christensen, L. Petrucci, Modular analysis of Petri nets, The Computer Journal 43 (2000) 224–242.

[11] L. Cabac, Modeling Petri Net-Based Multi-Agent Applications, Dissertation, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg, 2010. URL: https://ediss.sub.uni-hamburg.de/handle/ediss/3691.

[12] K. Jensen, Coloured Petri nets and the invariant-method, Theoretical computer science 14 (1981) 317–336.

[13] K. Jensen, How to find invariants for coloured Petri nets, in: International Symposium on Mathematical Foundations of Computer Science, Springer, 1981, pp. 327–338.

[14] C. Lakos, L. Petrucci, Modular state spaces and place fusion, in: International Workshop on Petri Nets and Software Engineering (PNSE 2007, associated with Petri Nets 2007), 2007, pp. 175–190.

[15] S. Christensen, L. Petrucci, Towards a modular analysis of coloured Petri nets, in: K. Jensen (Ed.), Application and Theory of Petri Nets 1992, 13th International Conference, Sheffield, UK, June 22-26, 1992, Proceedings, volume 616 of *Lecture Notes in Computer Science*, Springer, 1992, pp. 113–133. URL: https://doi.org/10.1007/3-540-55676-1_7.

[16] S. Christensen, L. Petrucci, Modular state space analysis of coloured Petri nets, in: International Conference on Application and Theory of Petri Nets, Springer, 1995, pp. 201–217.

[17] P. Fettke, W. Reisig, Handbook of Heraklit, 2021. Heraklit-working paper, v1.1, September 10, 2021, http://www.heraklit.org.

[18] W. Reisig, Simple composition of nets, in: G. Franceschinis, K. Wolf (Eds.), Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, June 22-26, 2009. Proceedings, volume 5606 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 23–42. URL: https://doi.org/10.1007/978-3-642-02424-5_4. doi:10.1007/978-3-642-02424-5_4.

[19] C. A. R. Hoare, Communicating sequential processes, Communications of the ACM 21 (1978) 666–677.

[20] I. Sakellariou, I. Vlahavas, I. Futo, Z. Pasztor, J. Szeredi, Communicating sequential processes for distributed constraint satisfaction, Information Sciences 176 (2006) 490–521.

[21] O. M. G. Inc., OMG Unified Modeling Language – version 2.5.1, https://www.omg.org/spec/UML/2.5.1, 2017. URL: https://www.omg.org/spec/UML/2.5.1, last accessed: 2024-06-05.

[22] L. Cabac, D. Moldt, Formal semantics for AUML agent interaction protocol diagrams, in: J. Odell, P. Giorgini, J. P. Müller (Eds.), The Fifth International Workshop on Agent-Oriented Software Systems (AOSE-2004). Proceedings, Columbia University, New York, USA, 2004, pp. 97–111. URL: http://dx.doi.org/10.1007/978-3-540-30578-1_4.

[23] D. Moldt, M. Hansson, L. Seifert, K. Ihlenfeldt, L. Clasen, K. Ehlers, M. Feldmann, Enriching heraklit modules by agent interaction diagrams, in: L. Gomes, R. Lorenz (Eds.), Application

and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings, volume 13929 of *Lecture Notes in Computer Science*, Springer Nature Switzerland AG, Cham, Switzerland, 2023, pp. 440–463. URL: https://doi.org/10.1007/978-3-031-33620-1_23. doi:10.1007/978-3-031-33620-1_23.

[24] F. A. Heitmann, Algorithms and hardness results for object nets, Ph.D. thesis, Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky, 2013.

[25] A. Syropoulos, Mathematics of multisets, in: Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View 1, Springer, 2001, pp. 347–358.

[26] O. Kummer, Referenznetze, Logos Verlag, Berlin, 2002. URL: http://www.logos-verlag.de/cgi-bin/engbuchmid?isbn=0035&lng=eng&id=.

[27] O. Kummer, F. Wienberg, M. Duvigneau, L. Cabac, M. Haustermann, D. Mosteller, Renew – the Reference Net Workshop, 2023. URL: http://www.renew.de/, release 4.1.

[28] A. T. Cohen, Data abstraction, data encapsulation and object-oriented programming, ACM SIGPLAN Notices 19 (1984) 31–35.

[29] W. v. d. Aalst, Verification of workflow nets, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, number 1248 in Lecture Notes in Computer Science, Springer Verlag, Berlin Heidelberg New York, 1997, pp. 407–426.

[30] K. Jensen, Coloured Petri Nets: Volume 1; Basic Concepts, Analysis Methods and Practical Use, EATCS Monographs on Theoretical Computer Science, Berlin Heidelberg New York, 1992.

[31] K. Jensen, L. M. Kristensen, Coloured Petri Nets - Modelling and Validation of Concurrent Systems, Springer, 2009. URL: https://doi.org/10.1007/b95112. doi:10.1007/b95112.

[32] C. Maier, D. Moldt, Object coloured Petri nets – A formal technique for object oriented modelling, in: B. Farwer, D. Moldt, M.-O. Stehr (Eds.), Report FBI-HH-B-205/97: Proceedings of the Workshop on Petri Nets in System Engineering (PNSE'97), Hamburg, September 25-26, 1997, number FBI-HH-B-205/97 in Report of the Department of Informatics, University of Hamburg, Department of Computer Science, 1997, pp. 11–19.

[33] C. Maier, D. Moldt, Dynamic structure and behaviour of coloured Petri nets supporting object-oriented modelling, in: W. van der Aalst, J.-M. Colom, F. Kordon, G. Kotsis, D. Moldt (Eds.), Petri Net Approaches for Modelling and Validation, LINCOM Studies in Computer Science, LINCOM Europa, München, 2002, pp. 81–101.

[34] M. Köhler, D. Moldt, H. Rölke, Modelling the structure and behaviour of Petri net agents, in: J. Colom, M. Koutny (Eds.), Proceedings of the 22nd Conference on Application and Theory of Petri Nets 2001, volume 2075 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001, pp. 224–241. URL: http://www.springerlink.com/link.asp?id=j4kbf32af81bba75.

[35] L. Cabac, M. Haustermann, D. Mosteller, Software development with Petri nets and agents: Approach, frameworks and tool set, Sci. Comput. Program. 157 (2018) 56–70. URL: https://doi.org/10.1016/j.scico.2017.12.003.

[36] J. R. Searle, Speech Acts, Cambride University Press, 1969.

[37] H. Rölke, Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen, volume 2 of *Agent Technology – Theory and Applications*, Logos Verlag, Berlin, 2004. URL: http://logos-verlag.de/cgi-bin/engbuchmid?isbn=0768&lng=eng&id=.

[38] J. Odell, H. van D. Parunak, B. Bauer, Extending UML for agents, in: G. Wagner, Y. Lesperance, E. Yu (Eds.), Agent-Oriented Information Systems. Workshop at the 17th National Conference on Artificial Intelligence (AAAI), AOIS 2000, 2000, pp. 3–17.

[39] K. Schmidt, Using Petri net invariants in state space construction, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2003, pp. 473–488.

[40] D. Moldt, J. Johnsen, R. Streckenbach, L. Clasen, M. Haustermann, A. Heinze, M. Hansson, M. Feldmann, K. Ihlenfeldt, RENEW: modularized architecture and new features, in: L. Gomes, R. Lorenz (Eds.), Application and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings, volume 13929 of *Lecture Notes in Computer Science*, Springer Nature Switzerland AG, Cham, Switzerland, 2023, pp. 217–228. URL: https://doi.org/10.1007/978-3-031-33620-1_12. doi:10.1007/978-3-031-33620-1_12.