Coupling and Cohesion Measures for Calculating P-Invariants for Modular P/T Nets*

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

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

Abstract

Decomposition of complex systems into smaller, independent modules is a fundamental approach for reducing complexity. The computation of structural invariants, a common method for verification, is aided by this modular reduction in complexity.

However, while T-invariants can be combined from the module's local T-invariants and local P-invariants are preserved, synchronization between modules introduces new P-invariants. Current algorithms do not consider these modular structures of the P/T nets for the calculations of P-invariants.

The incidence matrix of modular Petri nets, in which interactions between modules are restricted to only transitions, has a singly-bordered block diagonal (SBBD) structure. This structure may lead to speedups in reducing the matrix to row echelon form, the main challenge in the computation of invariants. In this paper, we examine the specific properties of modular nets that facilitate these speedups.

We present an algorithm for the reduction through Gaussian elimination. Through its analysis, we find that module size, number of external transitions and synchronized firing groups, and number of linearly independent internal transitions highly influence the runtime of our algorithm.

Our algorithm performs better than the naive Gaussian elimination for good modularized nets. The conditions leading to speed-ups naturally serve as formalizations of coupling and cohesion.

Keywords

Petri Nets, Invariants, Synchronous Channels, Modeling, Verification

1. Introduction

Design of complex and interacting systems was and still is a challenge. Modeling a system with specific perspectives is one way to address this. The divide-and-conquer principle has a long tradition in informatics, following far older sciences.

Parts of such decomposable systems are related in a specific way to specify the relationship. Parnas names such parts modules (see [1]). Nowadays the modules are also called components, services, entities, objects etc.

Fettke and Reisig use the notion of modules as their basic modeling entities [2] for their so called HERAKLIT approach. Petri nets can be used as a formal basis for the semantics of the modules. One central motivation for Fettke and Reisig is that Petri nets provide means to prove properties for the module models, even if they concentrate more on the understanding and modeling of systems in [2]. What is also very convincing of the HERAKLIT approach is the associative composition of modules that allows to build nested models of systems on an abstract level.

In [3] the HERAKLIT approach is combined with the ideas of agent-oriented modeling to emphasize the process perspective for entities that incorporate roles. Such roles are represented as agent templates. Their orchestration and choreography is modeled with an extended version of UML sequence charts.

Here we pick up the central ideas of modules to add some formal aspects to the models that can be created in the above contexts. One of the traditional concepts, invariants, are used to link modules. Calculation of invariants for very large Petri net models is difficult due to the algorithmic complexity resulting in long calculation times.

PNSE'25, International Workshop on Petri Nets and Software Engineering, 2025

*Supported by participants of our teaching project classes and student theses.

Based on the ideas of nets-within-nets [4] and modeling aspects of invariants [5] in combination with the decomposition of nets following the MULAN framework [6] we address the application of invariants to simple modular Petri net models. These modular Petri nets (P/T-nets) are coupled via synchronous channels to result in P/T-nets with synchronous channels [7]. [8] covered invariant calculations for P/T-Nets with synchronous channels. Here we extend all of the above aspects by a discussion of coupling and cohesion measures for calculating P-invariants for modular P/T Nets.

The following sections show the related work of the abovementioned topic Section 2 and the formal background of the paper Section 3. Requirements for the paper are presented in Section 4. Then, our algorithm for reducing a modular net's incidence matrix to row echelon form is given and explained in Section 5. In Section 6, an example run of the algorithm follows to give further illustration of its behavior, while the formal analysis is done in Section 7. Based on the findings in this analysis, we formally define notions of coupling and cohesion in Section 8 that naturally lead to formulating the modularization of P/T nets as the problem of permuting matrices into bordered block diagonal form. Then, we discuss our findings in Section 10 and conclude with Section 11.

2. Related Work

By dividing complex systems into different parts that exhibit significant internal behavior in comparison to the external interactions between each other, their subsequent analysis can be facilitated. Such a modular approach is realized in the context of Petri nets in the form of formal models as modular nets. In [9], such a model based on P/T nets was introduced. A modular P/T net consists of non-modular P/T nets serving as modules. Their interactions with each other are governed by fusion sets that allow sharing of places and synchronization of transitions. The characteristics of these modular P/T nets were then examined with respect to their P-invariants. A relation between local invariants and global invariants is given, but no explicit way of computing them is specified. In particular, it isn't further examined how specific modular structures could affect runtimes of such computations.

In [10, 11], modular CP-nets extend the capabilities of modular P/T nets by using colored Petri nets as modules. In this way, variables are allowed as edge weights through the use of synchronous channels. In the specific context of P/T nets, synchronous channels were integrated in [7] to define modular PTC-nets. In this way, variables as edge weights are possible while remaining close to the formal models of P/T nets. The modular PTC-nets were, however, strongly constrained in the possible synchronizations between modules.

In [8], the modular PTC-nets were expanded upon to give a more general model that isn't so heavily constrained called the PTC-system net. It also allows variables as edge weights and, together with the formulation through multisets, further increases capabilities of aggregating similar interface transitions into fewer external transitions. Both P-invariants and T-invariants were examined, but no way of computing them is explicitly given. It is this model that will serve as the formal basis for this paper. The properties of invariants in PTC-system nets were further examined in [12], where an algorithm for computing all of a net's P-invariants was given and analyzed, and which this paper will expand upon.

However, these papers only state in rough terms how the specifics of the modular structure actually aid in subsequent analysis. Especially, formal definitions of such measures of modularity could be of interest. The commonly used concepts of coupling and cohesion serve to describe the quality of modular systems and could also do so for modular nets.

In [13, 14], it was examined how to replace shared places by use of synchronous channels to generate more compact modular state spaces. It was found through experiments that weak coupling and strong cohesion were most beneficial, but these measures weren't expressed formally.

In [15], properties for potential measures of coupling and cohesion were formalized to enable precise statements about modular systems in the field of software engineering. They aren't specific measures but instead a general framework of how a measure should behave to be considered a coupling or cohesion measure.

Approaches to give specific measures of coupling and cohesion in accordance with this framework

include definitions based on information-theoretic principles [16, 17] and pertaining to different types of systems such as object-oriented systems [18] or knowledge-based systems [19]. Further formal definitions were posited based on a metamodel approach in [20].

In the context of Petri nets, formal measures were given in [21]. Among other things, the number of incoming arcs and the number of outgoing arcs were used as two separate bases to formulate coupling measures. However, these were only stated for a specific type of Petri net, and no measure for cohesion was given.

3. Preliminaries

3.1. Gaussian Elimination

Gaussian Elimination is an algorithm for solving systems of linear equations [22]. The algorithm transforms a given matrix, by swapping rows, adding rows or multiplying a row with a non-zero number. The resulting matrix is always in upper triangular form.

The algorithm eliminates all leading leftmost non-zero entries in each row but one, by adding multiples of the topmost row to the other rows, so that the given entry is zero. The rows are always sorted so that the number of leading zeros in each row is increasing. By iterating through these steps until there are no more than one leading non-zero entries per column, the matrix will be in row echelon form [22].

A matrix is in row echelon form when all rows containing only zeros are at the bottom and every leading non-zero entry is on the right of the leading non-zero entry of the row above. This implies that all entries under a leading coefficient are zeros [22].

$$\text{Original matrix } A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 5 & -3 \\ 3 & 6 & -2 \end{bmatrix}$$

$$\text{After Gaussian elimination (row echelon form): } \begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

For a given $n \times m$ matrix A, the number of leading non-zero entries in its row echelon form is referred to as the rank of A. It is denoted as $\mathrm{rank}(A)$ and always lies between 0 and $\min(n, m)$. Notably, $\mathrm{rank}(A) = \mathrm{rank}(A^{\mathsf{T}})$ and $\mathrm{rank}(A) = 0$ if and only if A only consists of zeros [23]. For the example above, from the row echelon form it is clear to see that $\mathrm{rank}(A) = 3$.

3.2. Bordered Block Diagonal Matrices

A matrix is in doubly bordered block diagonal form if it resembles this form:

$$A_{DB} = \begin{pmatrix} A_{11} & & & & C_1 \\ & A_{22} & & & C_2 \\ & & \dots & & \dots \\ & & & A_{nn} & C_n \\ R_1 & R_2 & \dots & R_n & E \end{pmatrix} \tag{1}$$

The blocks A_{ll} are matrices with the dimensions $n_l \times n_l$. The border blocks C_l and R_l have a dimension of $n_l \times p$ and $p \times n_l$, respectively, where ideally p should be significantly smaller than n_l [24]. Each column in the column border is called a coupling column, and each row in the row border is called a coupling row.

A singly bordered block diagonal matrix is a special arrangement of entries in a matrix:

Algorithm 1 Gaussian Elimination to row echelon form

Input: A matrix $A \in \mathbb{R}^{n \times m}$
Output: A matrix $\widetilde{A}$ in row echelon form
procedure GAUSSIAN ELIMINATION(A)
    for $col \in [0, m-1]$ **do**
        Identify the first non-zero entry in column col at or below the current row
        if entry found **then**
            Swap the current row with the row containing the first non-zero entry
        else
            continue ▷ Move to the next column
        end if

        Normalize the row containing the first non-zero entry, so that the leading entry becomes 1
        for each row below the current row **do**
            Subtract a multiple of the row containing the first non-zero entry to eliminate the entry in column col
        end for
    end for
    Rearrange all-zero rows to the bottom
end procedure

$$A_{SB} = \begin{pmatrix} A_{11} & & & & C_1 \\ & A_{22} & & & C_2 \\ & & \dots & & \dots \\ & & & A_{nn} & C_n \end{pmatrix} \tag{2}$$

The blocks A_{ll} are matrices with the dimensions $m_l \times n_l$. The border blocks C_l have a dimension of $m_l \times p$, where ideally p should be significantly smaller than n_l [24, 25].

3.3. P/T Nets with synchronous channels

Synchronous channels allow the linking of transitions which are part of the same net, or different nets. A transition which is part of a synchronous channel can either be a *Uplink* or a *Downlink*. In RENEW synchronous channels are defined with inscriptions. These inscriptions are either :<CHANNEL NAME>(<PARAMETERS>) for *Uplinks* or <NET-REFERENCE>:<CHANNEL NAME>(<PARAMETERS>) for *Downlinks*, as can be seen in Figure 1. Using the net-reference a synchronous channel can access another net with the same channel. Using the parameters variables can be passed through the channel. A transition can have multiple *Downlinks*, but only one *Uplink* [12].

Figure 1: Synchronous Channel Example, see [12]

3.4. Modular P/T nets

The basic idea behind the various different models for describing modular P/T nets [10, 7, 8] is to partition the elements of a P/T net into a set of disjoint P/T nets serving as modules. These modules each have internal behavior, but interactions between modules are governed by the overarching framework of the

modular P/T net. In the case of limiting interactions between modules to just transitions, this framework determines in which ways specific transitions of modules can be fired synchronously. Transitions that only fire when synchronized are called external transitions and constitute the interface of the module. Transitions that aren't external are called internal. In such a model, the internal transitions of modules behave as expected, while the synchronizations cause sets [10] or multisets [8] of external transitions to fire together.

The formal model serving as the basis for this paper will be the PTC-system net [8]. As such, it will also determine the notation used. A PTC-system net $\mathcal{SN}$ is a tuple $\mathcal{SN} = (S, K, R, Var, f_R)$, where S is a set of P/T-net modules, K is a set of channels, R is a set of synchronization rules, Var is a set of variables, and f_R is a variable assignment function.

The P/T-net modules $s \in S$ themselves are P/T nets with a designated set of external transitions T_{e_s}, which are each assigned a channel from K and potential assignments for variables f_s. Further, they allow variables to be used as arc weights. The synchronization rules are multisets over K and allow multisets of external transitions to fire if their channels sum to any such synchronization rule. At the same time, the variable assignment functions f_R and f_s ensure that all relevant variables are assigned a valid value. A multiset of external transitions that is allowed to fire in this way is called a synchronized firing group g, and the set of all firing groups is called G.

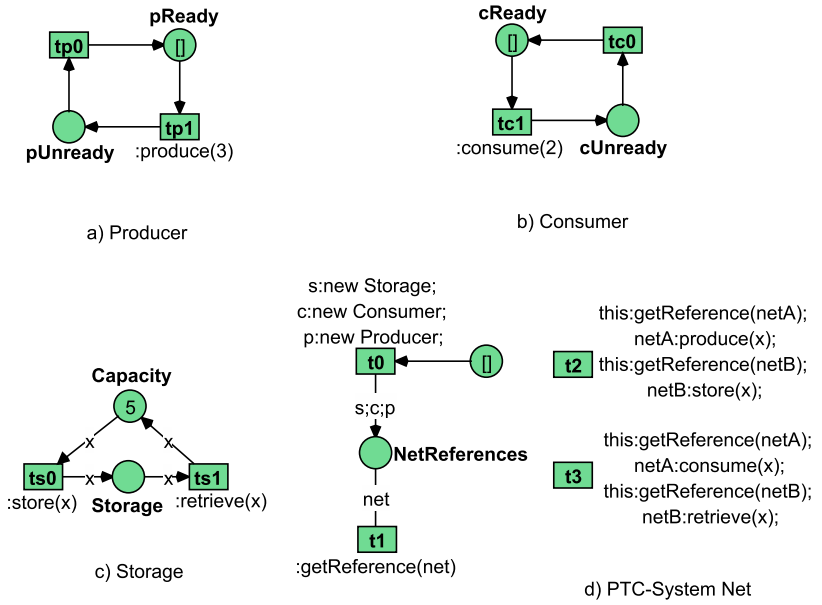

Figure 2: PTC-system net example, based on [8], with synchronization rules $\{\text{produce}, \text{store}\}_b$ and $\{\text{retrieve}, \text{consume}\}_b$

In figure 2, a simple example for a PTC-system net is given in RENEW. It consists of a producer and a consumer that are connected by a shared storage. All three are each a module in the PTC-system net, whose synchronization rules are expressed by the transitions **t2** and **t3**. The variable assignments are performed implicitly by the synchronous channels in RENEW. The synchronization rule $\{\text{produce}, \text{store}\}_b$ allows the firing group $\{\mathbf{tp1}, \mathbf{ts0}\}_b$ to fire while assigning x a value of 3. Similarly, $\{\text{retrieve}, \text{consume}\}_b$ allows $\{\mathbf{ts1}, \mathbf{tc1}\}_b$ to fire while assigning x a value of 2.

The change a synchronized firing group affects on specific places can then be expressed similarly to the arc weight function, leading to $\widetilde{W}_{sync} : (P \times G) \cup (G \times P) \to \mathbb{N}_0$. $\widetilde{W}_{sync}(p, g)$ describes how the marking of a place p is decreased if a firing group g fires, and $\widetilde{W}_{sync}(g, p)$ describes how it increases.

Together with the arc weight functions of all modules W_s, aggregated into a single function $\widetilde{W}$ with extended domain to include all places and transitions, the incidence matrix of a PTC-system net can be succinctly described. To better differentiate between internal and external components, the internal incidence matrix λ and the synchronized incidence matrix δ are introduced in addition to the global

incidence matrix Δ [12].

Definition 1. *The internal incidence matrix λ results from the effect of the internal transitions. Each module has an internal incidence matrix, which is calculated independently from the other modules. The entries of the internal incidence matrices aren't necessarily integers, because the external transition can include variables on their arc weights. These are only relevant if the whole net is considered.*

$$\lambda_{\mathcal{SN},pt} = \Delta t(p) = \widetilde{W}(t,p) - \widetilde{W}(p,t), \quad \textit{for } t \in T \setminus T_e \tag{3}$$

$$\lambda_{s,pt} = \Delta t(p) = \widetilde{W_s}(t,p) - \widetilde{W_s}(p,t), \quad \textit{for } t \in T_s \setminus T_{e_s},\ p \in P_s \tag{4}$$

Definition 2. *The synchronized incidence matrix δ considers the effects of synchronized transition sets. A synchronized incidence matrix exists for each module, in which the effects of the synchronized transition sets on the places of the modules are observed.*

$$\delta_{\mathcal{SN},pg} = \Delta g(p) = \widetilde{W}_{sync}(g,p) - \widetilde{W}_{sync}(p,g) \tag{5}$$

$$\delta_{s,pg} = \Delta g(p) = \widetilde{W}_{sync}(g,p) - \widetilde{W}_{sync}(p,g), \quad \textit{for } p \in P_s \tag{6}$$

Definition 3. *The global incidence matrix Δ results from the effects of the transition and synchronized transition sets from the PTC-System Net.*

$$\Delta_{\mathcal{SN},pt} = \Delta t(p) = \widetilde{W}(t,p) - \widetilde{W}(p,t), \quad \textit{for } t \in T \setminus T_e \tag{7}$$

$$\Delta_{\mathcal{SN},pg} = \Delta g(p) = \widetilde{W}_{sync}(g,p) - \widetilde{W}_{sync}(p,g) \tag{8}$$

The global incidence matrix can be formed with the internal incidence matrix λ and the synchronized incidence matrix δ. The resulting matrix is a singly bordered block diagonal matrix as described above. The rows and columns are sorted by the modules s_i.

$$\Delta = \begin{pmatrix} \lambda & \delta \end{pmatrix} = \begin{pmatrix} \lambda_{s_1} & 0 & \dots & \dots & 0 & \delta_{s_1} \\ 0 & \lambda_{s_2} & 0 & \dots & 0 & \delta_{s_2} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & \dots & 0 & \lambda_{s_{n-1}} & 0 & \delta_{s_{n-1}} \\ 0 & \dots & \dots & 0 & \lambda_{s_n} & \delta_{s_n} \end{pmatrix} \tag{9}$$

3.5. Formal Definitions of Coupling & Cohesion

As a foundation for discussing coupling and cohesion in modular systems we will use the formal definitions given in [15]. They are based on a representation of modular systems consisting of a set of elements and the relationships between these elements represented by a binary relation. Modules are then each a subset of these elements, collectively forming a partition. Relationships between elements are categorized as intra-module relationships IR connecting elements belonging to the same module and inter-module relationships connecting elements belonging to different modules. The latter are further divided into input and output relationships. However, for our purposes we will refer to these external relationships collectively as ER.

Intuitively, the coupling of a module describes how many relationships exist between it and other modules. The coupling of a modular system describes how many relationships exist between all its modules. Formally, the coupling of a module m or modular system MS is a function $\mathrm{coupling}(m)$ or $\mathrm{coupling}(MS)$ characterized by the following properties:

1. **Non-negativity**
   Coupling is non-negative:

$$\text{coupling}(m) \geq 0, \quad \text{coupling}(MS) \geq 0 \tag{10}$$

2. **Null Value**
   The coupling is 0 if the set of inter-module relationships in a module (ER_m) or in a modular system (ER) is empty:

$$ER_m = \emptyset \Rightarrow \text{coupling}(m) = 0, \quad ER = \emptyset \Rightarrow \text{coupling}(MS) = 0 \tag{11}$$

3. **Monotonicity**
   Adding inter-module relationships does not decrease coupling. Let m' be a module that results from a module m by adding inter-module relationships so that $ER_m \subseteq ER_{m'}$, and let MS' be a modular system that results from a modular system MS containing m by replacing m with m'. Then:

$$\text{coupling}(m) \leq \text{coupling}(m'), \quad \text{coupling}(MS) \leq \text{coupling}(MS') \tag{12}$$

4. **Merging of Modules**
   Merging two modules together does not increase coupling. Let m_1, m_2 be two modules, let m' be the union of m_1 and m_2, and let MS' be a modular system that results from a modular system MS containing m_1, m_2 by replacing both with m'. Then:

$$\text{coupling}(m_1) + \text{coupling}(m_2) \geq \text{coupling}(m'), \quad \text{coupling}(MS) \geq \text{coupling}(MS') \tag{13}$$

5. **Disjoint Module Additivity**
   In the case of two unrelated modules being merged, the coupling remains unchanged. If there are no relationships between m_1 and m_2, then:

$$\text{coupling}(m_1) + \text{coupling}(m_2) = \text{coupling}(m'), \quad \text{coupling}(MS) = \text{coupling}(MS') \tag{14}$$

Intuitively, the cohesion of a module describes how well the elements of the module are grouped together. The cohesion of a modular system acts as a comprehensive measure for the cohesion of all its modules. Formally, the cohesion of a module m or modular system MS is a function $\text{cohesion}(m)$ or $\text{cohesion}(MS)$ characterized by the following properties:

1. **Non-negativity and Normalization**
   Cohesion is non-negative and normalized:

$$\text{cohesion}(m) \in [0, 1], \quad \text{cohesion}(MS) \in [0, 1] \tag{15}$$

2. **Null Value**
   The cohesion is 0 if the set of intra-module relationships in a module (IR_m) or in a modular system (IR) is empty:

$$IR_m = \emptyset \Rightarrow \text{cohesion}(m) = 0, \quad IR = \emptyset \Rightarrow \text{cohesion}(MS) = 0 \tag{16}$$

3. **Monotonicity**
   Adding intra-module relationships does not decrease cohesion. Let m' be a module that results from a module m by adding intra-module relationships so that $IR_m \subseteq IR_{m'}$, and let MS' be a modular system that results from a modular system MS containing m by replacing m with m'. Then:

$$\text{cohesion}(m) \leq \text{cohesion}(m'), \quad \text{cohesion}(MS) \leq \text{cohesion}(MS') \tag{17}$$

4. **Cohesive Modules**
   Merging together unrelated modules does not increase cohesion. Let m_1, m_2 be two modules that have no relationships between them, i.e., $ER_{m_1} \cap ER_{m_2} = \emptyset$, let m' be the union of m_1 and m_2, and let MS' be a modular system that results from a modular system MS containing m_1, m_2 by replacing both with m'. Then:

$$\max(\text{cohesion}(m_1), \text{cohesion}(m_2)) \geq \text{cohesion}(m'), \quad \text{cohesion}(MS) \geq \text{cohesion}(MS') \tag{18}$$

4. Problem Description

Given the incidence matrix Δ of a P/T net, the net's T-invariants are solutions of the homogenous linear system $\Delta \boldsymbol{x} = \mathbf{0}$ and the P-invariants are solutions of $\Delta^{\mathsf{T}} \boldsymbol{x} = \mathbf{0}$. The computation of invariants is based on solving a linear system $A\boldsymbol{x} = \mathbf{0}$.

This is done through two steps. First, the matrix A is reduced to row echelon form $\widetilde{A}$. Then, the reduced linear system $\widetilde{A}\boldsymbol{x} = \mathbf{0}$ is solved through backward substitution. Between these two steps, it is the reduction to row echelon form that is more computationally expensive [26]. Therefore, our goal is to exploit the inherent modular structure of modular P/T nets to improve this reduction over the direct application of Gaussian elimination.

Further, for modular nets with synchronous channels, the net's T-invariants can be combined from local T-invariants through their coupling components [27]. Because of this, it isn't necessary to reduce Δ in its entirety. This is in contrast to P-invariants, where local invariants are preserved even when considering the entire modular net. However, through the interactions between modules, further P-invariants can emerge [8]. To compute these invariants, the net has to be considered in its entirety. What is, therefore, required is a reduction of Δ^{T} specifically.

5. Modular Gaussian Elimination

In this section we describe an algorithm for reducing an incidence matrix Δ^{T} to row echelon form. We will first give an overview of the entire algorithm. Then, by dividing the algorithm into three phases, we will describe each part in more detail. Since we only consider the transposed incidence matrix, we will refer to the transpose matrices without the superscript T from this point forward, keeping notation simple. So, for example, instead of $\Delta^{\mathsf{T}}, \lambda^{\mathsf{T}}, \delta^{\mathsf{T}}$, we will just call them Δ, λ, δ.

Reduction of the incidence matrix Δ can be divided into two parts. Firstly, the reduction of its block diagonal part λ, where care is taken to ensure properties that aid in the following part. And secondly, reduction of the block border part δ, using the already reduced λ.

5.1. Overview

The modular Gaussian elimination, given in Algorithm 2, takes an incidence matrix Δ consisting of an internal incidence matrix λ and a synchronized incidence matrix δ. As λ is a block diagonal matrix, Δ is in SBBD form:

$$\Delta = \begin{pmatrix} \lambda \\ \delta \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_k \\ \delta_1 & \delta_2 & \cdots & \delta_k \end{pmatrix} \tag{19}$$

As the blocks in λ are independent of each other, the main challenge in reducing Δ is introduced by the interactions between modules contained in δ. First, this independence is completely utilized in **Phase 1** to reduce just λ. Then, it is used as much as possible in **Phase 2** to eliminate entries in δ.

This is done by using rows from the just reduced λ to avoid interactions between modules. However, in general, such an isolated approach doesn't suffice so that some entries in δ remain. Finally, these remaining entries are eliminated in **Phase 3**. This requires using rows from δ itself so that interactions between modules are unavoidable. As a last step, simple row permutations are performed so that a proper row echelon form is achieved.

Algorithm 2 Reduction of Δ to row echelon form

Input: A matrix $\Delta = \begin{pmatrix} \lambda \\ \delta \end{pmatrix} \in \mathbb{Z}^{|T|\times|P|}$ in SBBD form
Output: The matrix $\widetilde{\Delta}$ in row echelon form
procedure MODULAR GAUSSIAN ELIMINATION(A)
    // **Phase 1:** Reduction of block diagonal matrix $\lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$
    for all $i \in [1, k]$ **do**
        Transform λ_i to row echelon form $\widetilde{\lambda_i}$
        Move leading entries to the main diagonal by swapping columns
    end for

    // **Phase 2:** Initial reduction of synchronization border
    for all $i \in [1, k]$ **do**
        for all $j \in [1, \mathrm{rank}(\widetilde{\lambda_i})]$ **do**
            Use row j of $\widetilde{\lambda_i}$ to eliminate all entries in column j of δ_i forming δ'
        end for
    end for
    // **Phase 3:** Final reduction of synchronization border
    Eliminate remaining entries in δ' transforming it to row echelon form $\widetilde{\delta}$
    Rearrange rows in $\begin{pmatrix} \widetilde{\lambda} \\ \widetilde{\delta} \end{pmatrix}$ to get row echelon form $\widetilde{\Delta}$
end procedure

5.2. Phase 1: Reduction of block diagonal matrix

Due to the block diagonal structure of λ, each of its blocks can be considered in isolation. We reduce each individual block λ_i into row echelon form through Gaussian. Then, the resulting matrix is transformed through column permutations to ensure that each leading entry lies along the main diagonal. We will call the thusly transformed matrix $\widetilde{\lambda_i}$.

The same column permutations are also performed on δ, keeping the columns between both parts of Δ consistent. Since these permutations are a simple rearrangement of the columns of δ, we won't introduce a specific designator and will keep referring to it as δ.

The permutations performed on each λ_i are illustrated by taking matrix R in row echelon form as an example. The leading entries in rows 3 and 4 do not lie on the main diagonal but are located in columns 5 and 7, respectively. To achieve the desired structure, columns 3 and 5 are swapped, as well as columns 4 and 7. The resulting matrix $\widetilde{R}$ now has all its leading entries along its main diagonal.

$$R = \begin{pmatrix} 1 & 5 & 2 & 1 & 0 & 5 & 4 & 0 \\ 0 & 3 & 2 & 3 & 2 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 4 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \widetilde{R} = \begin{pmatrix} 1 & 5 & 0 & 4 & 2 & 5 & 1 & 0 \\ 0 & 3 & 2 & 0 & 2 & 1 & 3 & 2 \\ 0 & 0 & 1 & 3 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{20}$$

Having reduced every block λ_i to $\widetilde{\lambda_i}$ in this way, we get the reduced $\widetilde{\lambda}$ as a block diagonal matrix

with all individual reduced blocks along its diagonal:

$$\widetilde{\lambda} = \begin{pmatrix} \widetilde{\lambda_1} & 0 & \cdots & 0 \\ 0 & \widetilde{\lambda_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \widetilde{\lambda_k} \end{pmatrix} \tag{21}$$

Note that $\widetilde{\lambda}$ is only almost in row echelon form as it may contain rows with all entries equal to zero not located at the bottom of the matrix. This is due to the zero rows of a diagonal block being located higher than the entries of the following blocks. The following matrix illustrates this for the case $k = 2$ with two 3×2 submatrices:

$$\begin{pmatrix} * & * & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{22}$$

These zero rows could be permuted to the bottom of the block diagonal matrix to achieve a proper row echelon form. This form is, in a sense, equivalent to row echelon form with respect to simple row permutations. Therefore, this form of reduction suffices to continue with the subsequent phase.

5.3. Phase 2: Initial reduction of synchronization border

After the reduction of λ, the incidence matrix Δ has been transformed to:

$$\begin{pmatrix} \widetilde{\lambda} \\ \delta \end{pmatrix} = \begin{pmatrix} \widetilde{\lambda_1} & 0 & \cdots & 0 \\ 0 & \widetilde{\lambda_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \widetilde{\lambda_k} \\ \delta_1 & \delta_2 & \cdots & \delta_k \end{pmatrix} \tag{23}$$

While the λ blocks were entirely independent of each other, the same does not apply to δ, which relates to the synchronization between modules. However, by using $\widetilde{\lambda_i}$ for eliminating entries in δ_i, the corresponding row additions only affect the columns of module i. In this way, independence between modules can be further exploited.

Due to the construction of $\widetilde{\lambda_i}$, all its leading entries lie along its main diagonal and the rest of the main diagonal is filled with zeros. Therefore, the number of these entries is equal to the rank of the matrix $\mathrm{rank}(\widetilde{\lambda_i})$. Each corresponding row allows elimination of all entries in δ_i located in the same column as the leading entry. Through these eliminations, δ_i is transformed into δ_i' in which the first $\mathrm{rank}(\widetilde{\lambda_i})$ columns' entries are all equal to zero.

$$\begin{pmatrix} \widetilde{\lambda_i} \\ \delta_i \end{pmatrix} = \left(\begin{array}{ccccc} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right) \quad \longrightarrow \quad \begin{pmatrix} \widetilde{\lambda_i} \\ \delta_i' \end{pmatrix} = \left(\begin{array}{ccccc} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & * & * \end{array}\right) \tag{24}$$

In the context of modular P/T nets, this process can be further simplified. The rows in δ are the effects of the synchronized firing groups, which result from the combined effects of the participating external transitions. If the entry for a specific column has been eliminated for each external transition, meaning that it is then equal to 0, any way of combining such transitions will always yield an effect with that specific entry also being equal to 0. Therefore, it suffices to reduce just these external transitions for each module.

If the elimination in **Phase 2** is done on the external transitions instead of the synchronized firing groups, in which all variables already have a unique value assigned to them, the question arises how to handle variables. One approach would be to just leave them as is and to perform the elimination steps through symbolic computations. However, such computations are inherently more expensive than numerical computations and, further, could make additional entries depend on the specific value of the variable.

Indicating against the unconstrained use of variables is the fact that all external transitions of a module can be replaced by a single transition if assigning negative values is permissible. This is illustrated in Figure 3, where assignments of x and y with 1 and 3, 2 and 1, and 1 and 0 allow emulating each of the three replaced transitions. It would then be unreasonable to expect that a problem of any size could be reduced to just a constant size.

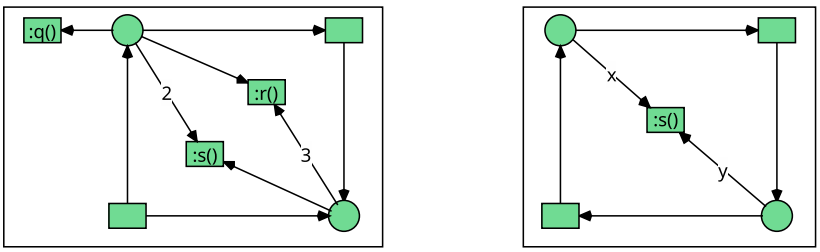

Figure 3: Three external transitions are replaced by a single external transition using variables

However, due to only partial reductions being performed in **Phase 2**, variables can be effectively hidden from the numerical computations being performed. If a variable of a module i is located in one of the last $|P_i| - \operatorname{rank}(\widetilde{\lambda}_i)$ columns, i.e., a column whose entries won't be eliminated in **Phase 2**, then the symbolic value of its entry at the end of the phase will be equal to the variable itself plus some numerical value. As such, the variable can simply be treated as 0 during computations, and only in the instantiation of the synchronized firing groups for **Phase 3** will the variable be assigned its value. The computed numerical value then simply serves as an offset in this assignment.

5.4. Phase 3: Final reduction of synchronization border

In the case that $\widetilde{\lambda}_i$ has rank lower than its number of columns, some entries in δ_i remain, as exemplified in Equation (24). These entries require rows in δ itself to be used for elimination. However, such rows are not contained to any specific module but instead include nonzero entries in columns belonging to modules that participate in the synchronized firing group represented by that row.

Without presupposing anything about anything about the specific external transitions and participation of modules in synchronizations, all entries not yet eliminated could potentially be a nonzero entry. As such, each block δ'_i consists of some initial zero columns followed by the remaining columns filled with arbitrary entries. The following matrix exemplifies this structure for δ':

$$\delta' = \begin{pmatrix} \delta'_1 & \delta'_2 & \delta'_3 & \cdots & \delta'_k \end{pmatrix} = \left(\begin{array}{cc:ccc:cc:c:cccc} * & * & 0 & 0 & * & 0 & 0 & & 0 & 0 & * & * \\ * & * & 0 & 0 & * & 0 & 0 & & 0 & 0 & * & * \\ * & * & 0 & 0 & * & 0 & 0 & \cdots & 0 & 0 & * & * \\ * & * & 0 & 0 & * & 0 & 0 & & 0 & 0 & * & * \\ * & * & 0 & 0 & * & 0 & 0 & & 0 & 0 & * & * \end{array}\right) \tag{25}$$

δ' is then reduced to row echelon form by Gaussian elimination, transforming it into $\widetilde{\delta}$. Every row of $\widetilde{\delta}$ that was used to eliminate entries in the rows below it will contain entries that can't be further eliminated. When considering Δ in its entirety, it is therefore not yet in row echelon form. However, by either swapping these rows with zero rows of the $\widetilde{\lambda}$, which columns its leading entry belongs to, or by inserting it below $\widetilde{\lambda}$ if no such zero row exists, full row echelon form of Δ is achieved.

6. Computation Example of the Algorithm

As an example to illustrate how the algorithm works, we will consider the modular net in Figure 4. It consists of 4 modules that are synchronized by two synchronization rules, resulting in 4 possible synchronized firing groups.

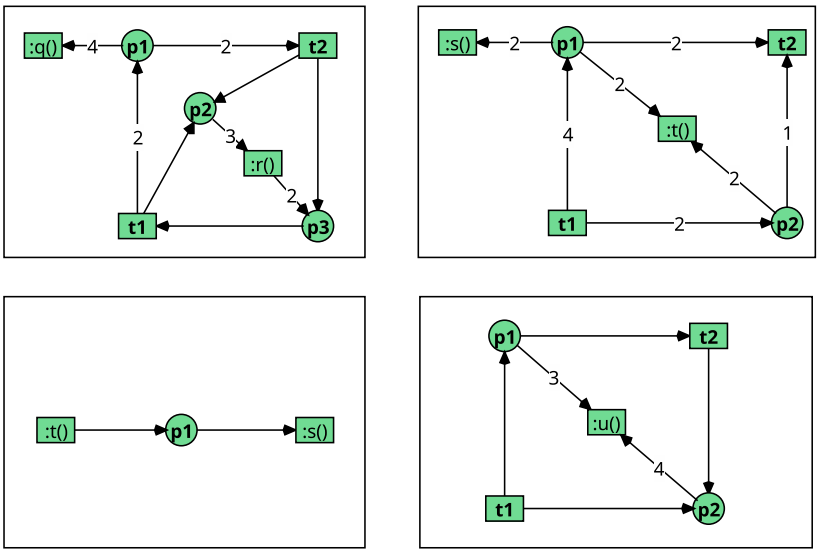

Figure 4: 4 modules synchronized by the synchronization rules $\{r, t, u\}_b$ and $\{q, s, u, u\}_b$

The net's modular structure results in the following incidence matrix Δ:

$$\Delta = \left(\begin{array}{ccc:ccc:cc} 2 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -1 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ \hline 0 & -3 & 2 & -2 & -2 & 0 & -3 & -4 \\ 0 & -3 & 2 & 0 & 0 & 1 & -3 & -4 \\ -4 & 0 & 0 & -2 & 0 & 0 & -6 & -8 \\ -4 & 0 & 0 & 0 & 0 & -1 & -6 & -8 \end{array}\right) \tag{26}$$

The dashed lines indicate the individual modules, with vertical lines separating the places and horizontal lines separating the internal transitions. The solid line divides Δ into the internal incidence matrix λ in the upper half and the synchronized incidence matrix δ in the lower half.

The modules are arranged in natural reading order. For example, the first block contains the internal incidence matrix λ_1 of the upper left module. It consists of the effect of its internal transition **t1** as the first row and the effect of **t2** as the second row. Notably, the third module doesn't contain any internal transitions so that its internal incidence matrix is empty.

The first and second rows in δ both result from the synchronization rule $\{r, t, u\}_b$. Multiple external transitions inscribed with the channel t existing, one in the upper right and the other in the lower left module, lead to a choice in synchronization. Consequently, two different synchronized firing groups

are possible. In the first row, the transition from the upper right module is chosen, and in the second row, the transition from the lower left. The third and fourth rows, resulting from $\{q, s, u, u\}_b$, behave similarly based on the choice regarding channel s.

In **Phase 1** of the algorithm, each individual block λ_i is reduced:

$$\lambda_1 = \begin{pmatrix} 2 & 1 & -1 \\ -2 & 1 & 1 \end{pmatrix}, \lambda_2 = \begin{pmatrix} 4 & 2 \\ -2 & -1 \end{pmatrix}, \lambda_4 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \quad \longrightarrow \quad \widetilde{\lambda_1} = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 2 & 0 \end{pmatrix}, \widetilde{\lambda_2} = \begin{pmatrix} 4 & 2 \\ 0 & 0 \end{pmatrix}, \widetilde{\lambda_4} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \tag{27}$$

Then, in **Phase 2**, each reduced $\widetilde{\lambda_i}$ is used to eliminate entries in the corresponding δ_i. Instead of doing this for every synchronized firing group, it suffices to do so for every external transition, as the former are composed of the latter.

$$\begin{pmatrix} \widetilde{\lambda_1} \\ \delta_1 \end{pmatrix} = \left(\begin{array}{ccc} 2 & 1 & -1 \\ 0 & 2 & 0 \\ \hline 0 & -3 & 2 \\ -4 & 0 & 0 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_2} \\ \delta_2 \end{pmatrix} = \left(\begin{array}{cc} 4 & 2 \\ 0 & 0 \\ \hline -2 & -2 \\ -2 & 0 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_3} \\ \delta_3 \end{pmatrix} = \left(\begin{array}{c} 1 \\ \hline -1 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_4} \\ \delta_4 \end{pmatrix} = \left(\begin{array}{cc} 1 & 1 \\ 0 & 2 \\ \hline -3 & -4 \end{array}\right) \tag{28}$$

$$\longrightarrow \quad \begin{pmatrix} \widetilde{\lambda_1} \\ \delta_1' \end{pmatrix} = \left(\begin{array}{ccc} 2 & 1 & -1 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 2 \\ 0 & 0 & -2 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_2} \\ \delta_2' \end{pmatrix} = \left(\begin{array}{cc} 4 & 2 \\ 0 & 0 \\ \hline 0 & -1 \\ 0 & 1 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_3} \\ \delta_3' \end{pmatrix} = \left(\begin{array}{c} 1 \\ \hline -1 \end{array}\right), \begin{pmatrix} \widetilde{\lambda_4} \\ \delta_4' \end{pmatrix} = \left(\begin{array}{cc} 1 & 1 \\ 0 & 2 \\ \hline 0 & 0 \end{array}\right) \tag{29}$$

Combining these reduced external transitions according to the synchronization rules results in the partially reduced δ'. Finally, δ' is reduced to $\widetilde{\delta}$ in **Phase 3**:

$$\delta' = \begin{pmatrix} 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & -1 & 0 & 0 \end{pmatrix} \quad \longrightarrow \quad \widetilde{\delta} = \begin{pmatrix} 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{30}$$

Together, $\widetilde{\lambda}$ and $\widetilde{\delta}$ are then almost in row echelon form so that the rows of $\widetilde{\delta}$ simply have to be inserted in $\widetilde{\lambda}$:

$$\left(\begin{array}{cccc:cccc} 2 & 1 & -1 & & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & & 4 & 2 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & & 0 & 0 & 0 & 2 \\ \hline 0 & 0 & 2 & & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 0 & 0 \end{array}\right) \rightsquigarrow \begin{pmatrix} 2 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{31}$$

As can be seen, due to $rank(\widetilde{\lambda_1}) = 2$ being smaller than $|P_1| = 3$, a row with a nonzero entry in the third column remains in $\widetilde{\delta}$, requiring insertion. At the same time, $\widetilde{\lambda_1}$ having more columns than rows means that there is space below it in regard to the row echelon form. This space, then, allows the row from $\widetilde{\delta}$ to be inserted as the third row.

With $rank(\widetilde{\lambda_2}) = 1 < |P_2| = 2$, the same applies, resulting in the second row of $\widetilde{\delta}$ being inserted below $\widetilde{\lambda_2}$. In this case, a zero row has to be swapped to the bottom of the entire matrix in return.

While $rank(\widetilde{\lambda_3}) = 0 < |P_3| = 1$ implies that there is a third row in $\widetilde{\delta}$ to be expected, this row happened to be eliminated in the process of reducing δ' to $\widetilde{\delta}$. This fact also means that an invariant exists in the whole modular net. Specifically, backward substitution yields the following P-invariant:

$$\begin{pmatrix} -1 & 0 & -2 & 2 & -4 & 4 & 0 & 0 \end{pmatrix}^\mathsf{T} \tag{32}$$

7. Analysis

In this section, we will shortly justify that Algorithm 2 works and terminates correctly. Then, we will analyse the runtime of each of the algorithm's phases before giving a combined overview.

7.1. Correctness and Termination

The Algorithm 2 works mostly like normal Gaussian elimination by using elementary row operations. It deviates by changing the order in which entries are eliminated and by performing column permutations in **Phase 1**.

The change in order results from performing operations in the outlined phases instead of simply going row by row. However, this order serves to describe the algorithm more clearly and to ensure constraints on the row permutations performed. These constraints do not limit the correctness of the elimination, as they do not prevent the choice of pivot elements if no other alternatives exist. Such a choice is just postponed to **Phase 3**.

Using column permutation is unproblematic, as it does not interfere with the row permutations necessary for selecting pivot elements. Further, they correspond to changing the order of variables in the linear system. When subsequently solving the linear system, the column permutations simply have to be performed in reverse order on the solution vectors to get the correct arrangements of variables.

Overall, therefore, the correctness and termination of Algorithm 2 follow directly from that of the Gaussian elimination.

7.2. Runtime of Phase 1: Reduction of block diagonal matrix

When only considering the internal elements of the individual modules, it is to be expected that computations involving the entire modular net can be reduced to addressing each module individually. Just that is what we find for the reduction of the block diagonal matrix to row echelon form. Each block λ_i is reduced separately so that the runtime of reducing λ doesn't depend on the modular net's size as a whole but on the individual sizes of its modules.

Since the algorithm doesn't depend on the specific method used for reducing each λ_i to row echelon form, we simply assume it to be an arbitrary runtime $T(|P_i|, |T_i|)$ depending on the corresponding module's number of places and transitions. In the simplest case, this would be Gaussian elimination with $T(|P_i|, |T_i|) = \mathcal{O}\left(\max(|P_i|, |T_i|) \min(|P_i|, |T_i|)^2\right)$ [28]. But more efficient algorithms exist, such as Strassen's algorithm with $T(n) = \mathcal{O}\left(n^{\log_2(7)}\right) \approx \mathcal{O}\left(n^{2.8}\right)$ for $n \times n$ matrices [29, 30], methods for significant differences in $|P_i|$ and $|T_i|$ [31, 32], or potential optimizations based on the specific sparse structure of the module. This way, it follows for the combined runtime of all modules:

$$\mathcal{O}\left(\sum_{i=1}^{k} T(|P_i|, |T_i|)\right) \tag{33}$$

Even though the sum of maximum terms can be higher than the maximum of the entire net, i.e., $\sum_{i=1}^{k} \max(|P_i|, |T_i|) \geq \max(|P|, |T|)$, in terms of time complexity, the more influential minimum terms are lower, i.e., $\sum_{i=1}^{k} \min(|P_i|, |T_i|) \leq \min(|P|, |T|)$. It follows then that modules that deviate strongly from being square matrices are beneficial for the block diagonal matrix reduction. In the following, it is therefore justified to assume $n_i = |P_i| = |T_i|$ and let $n = n_1 + n_2 + \cdots + n_k$. Expressing the runtime as a polynomial n^ω with $\omega \geq 2$ leads to the following inequality:

$$\sum_{i=1}^{k} n_i^\omega = \left(\left(\sum_{i=1}^{k} |n_i|^\omega\right)^{\frac{1}{\omega}}\right)^\omega = (\|(n_1, n_2, \cdots, n_k)\|_\omega)^\omega \leq (\|(n_1, n_2, \cdots, n_k)\|_1)^\omega = \left(\sum_{i=1}^{k} |n_i|\right)^\omega = n^\omega \tag{34}$$

The inequality follows from the relations between different p-norms, where $\|\boldsymbol{x}\|_q \leq \|\boldsymbol{x}\|_p$ for all $1 \leq p \leq q < \infty$ [33]. This confirms the intuitive notion that it is indeed more efficient to solve k independent subproblems than to solve the entire problem at once. Moreover, the inequality is strict if there is more than one non-empty module.

However, in terms of asymptotic runtime, the size of the largest module dominates. This leads to the first condition for good modularization, requiring modules to all be of roughly the same size n_s so that $n_s \approx n/k$. In that case, the overall runtime simplifies to:

$$\mathcal{O}\left(\sum_{i=1}^{k} n_i^{\omega}\right) \approx \mathcal{O}\left(k n_s^{\omega}\right) = \mathcal{O}\left(k^{(1-\omega)} n^{\omega}\right) \tag{35}$$

Consequently, if the number of modules k grows with the overall size of the system, a better asymptotic runtime is achieved. For naive Gaussian elimination, where $\omega = 3$, this translates to an improvement by a factor of $\frac{1}{k^2}$. The realistic best case is achieved if the modules are of constant size so that $k \in \mathcal{O}(n)$. Irrespective of the actual algorithm used as a subroutine, the block diagonal matrix would then be reduced to row echelon form in time $\mathcal{O}(n)$.

Beyond these runtime improvements, it is notable that the reduction of the individual blocks is fully parallelizable and that modules with the same incidence matrix only require the reduction to be computed once instead of for each of those modules.

7.3. Runtime of Phase 2: Initial reduction of synchronization border

In total, the reduction of δ to δ' involves the elimination of all entries in the first $\mathrm{rank}(\widetilde{\lambda_i})$ columns in δ_i for each i. And for each column, the number of entries to be eliminated is equal to the number of rows in δ_i, which is equal to $|G|$. Each elimination of an entry in δ_i using the j-th row from $\widetilde{\lambda}$ involves adding a multiple of a row with $|P_i| - j$ entries. Consequently, the reduction in its entirety requires a number of operations in the order of

$$\mathcal{O}\left(\sum_{i=1}^{k} \mathrm{rank}(\widetilde{\lambda_i})|G||P_i|\right) \tag{36}$$

However, because each elimination step only affects the entries in δ_i, its rows contain duplicate entries if external transitions participate in more than just one synchronization each. Therefore, instead of $|G|$, the effective number of rows becomes $|T_e|$ and the runtime is reduced to

$$\mathcal{O}\left(\sum_{i=1}^{k} \mathrm{rank}(\widetilde{\lambda_i})|T_e||P_i|\right) \tag{37}$$

In both cases, the runtime depends on the size of the modules by way of the number of their respective places. Note that $\mathrm{rank}(\widetilde{\lambda_i})$ also depends on this notion of size as it can only take on values from 1 to $|P_i|$. Therefore, the same considerations concerning module size apply here as for the block diagonal matrix. For modules of roughly the same size ($|P_s| \approx |P|/k$), the runtime simplifies to:

$$\mathcal{O}\left(\sum_{i=1}^{k} \mathrm{rank}(\widetilde{\lambda_i})|T_e||P_i|\right) \approx \mathcal{O}\left(|T_e||P_s|^2 \sum_{i=1}^{k} \frac{\mathrm{rank}(\widetilde{\lambda_i})}{|P_i|}\right) = \mathcal{O}\left(\frac{1}{k^2}|T_e||P|^2 \sum_{i=1}^{k} \frac{\mathrm{rank}(\widetilde{\lambda_i})}{|P_i|}\right) \tag{38}$$

Because $\mathrm{rank}(\widetilde{\lambda_i})$ can only take on values from 1 to $|P_i|$, the summands can be upwardly bounded by 1, leading to the following simplification:

$$\mathcal{O}\left(\frac{1}{k^2}|T_e||P|^2 \sum_{i=1}^{k} 1\right) = \mathcal{O}\left(\frac{1}{k}|T_e||P|^2\right) \tag{39}$$

It follows then that for modules of constant size ($k \in \mathcal{O}(|P|)$), the reduction of δ to δ' is performed in time $\mathcal{O}(|T_e||P|)$. Just as for the block diagonal matrix, this reduction is fully parallelizable, and modules that have the same external transitions in addition to having the same internal incidence matrix don't require separate computations.

7.4. Runtime of Phase 3: Final reduction of synchronization border

Through the initial reduction, from the original $|P_i|$ columns in δ'_i, $\mathrm{rank}(\widetilde{\lambda_i})$ have had all their entries eliminated. Excluding columns with only zeros, the effective number of columns c is therefore:

$$c = \sum_{i=1}^{k}(|P_i| - \mathrm{rank}(\widetilde{\lambda_i})) = |P| - \mathrm{rank}(\widetilde{\lambda_i}) = |P| - \mathrm{rank}(\lambda) \tag{40}$$

Consequently, δ' is effectively a $|G| \times c$ matrix. As was the case for the blocks in λ, the subsequent reduction of δ' can be performed by any algorithm for reducing matrices to row echelon. In the case of naive Gaussian elimination, this leads to the following runtime:

$$\mathcal{O}\left(\max(c, |G|)\min(c, |G|)^2\right) \tag{41}$$

This means that these computations, in contrast to the previous ones, cannot be considered as independent computations for each module. The runtime of the entire algorithm is therefore dominated by the reduction from δ' to $\widetilde{\delta}$. What is gained by the previous two reductions is therefore an effective reduction in rows and columns. The overall $|T \setminus T_e| + |G|$ rows were reduced to just the $|G|$ rows corresponding to the interactions between modules, and the $|P|$ columns were reduced to just a fraction depending on the rank of the internal incidence matrix.

7.5. Overall runtime

Both **Phase 1** and **Phase 2** make use of the modular structure in such a way that their runtimes are significantly improved if the modular net is divided into modules of appropriate size. Additionally, **Phase 2** is influenced by the size of the interfaces between modules, measured by $|G|$ or more specifically by $|T_e|$. On the other hand, **Phase 3** doesn't benefit in the same way so that its runtime grows asymptotically the most with the overall size of the modular net. Besides the number of places, the runtime also increases with the size of the interfaces between modules $|G|$. To some degree these costly elimination steps, which involve the coupling components between modules, can be alleviated by work done in **Phase 2**, exploiting the independence between modules. The extent of this improvement results from the rank of the internal incidence matrix $\mathrm{rank}(\lambda)$.

These measures directly influence the overall runtime, and as such, they serve as indicators for how the modularity of a given modular net aids in its analysis. In this sense, they determine whether a modular net has good or bad modularity.

8. Coupling & Cohesion in Modular P/T Nets

The runtime analysis in Section 7 yielded formal expressions for assessing the modularity of a modular P/T net. We aim to put these indicators into the proper context by classifying what kind of measures they are. Specifically, we will express them as measures for coupling and cohesion.

However, we will first reconsider the underlying framework for describing the modular system. In [15], the elements of a modular system are connected by a binary relation. In the context of P/T nets, it is natural to let both places and transitions be elements in this sense. These elements are related to each other if they are connected by the flow relation F.

While such an approach can most definitely be taken [21], we find that in the context of modularization through synchronous channels, it is more appropriate to have just the places as elements and the transitions serve as relationships between these elements. Instead of the modular system being described by a directed graph, it is then a hypergraph with places as vertices and transitions as hyperedges. In this way, the modularization through synchronous channels corresponds to a separation of the elements, with just the relationships between these elements serving as connections between modules. The categorization of relationships into intra-module IR and inter-module ER follows naturally. Transitions that are only connected to places of a single module are intra-module relationships, and all

other transitions are inter-module relationships. With this basis, we can now formally define coupling and cohesion.

Both the runtime of **Phase 2**, as given in Equation (36), and **Phase 3**, given in Equation (41), depend on the number of synchronized firing groups $|G|$. On this basis, we formulate the first definition of coupling.

Definition 4. *The coupling of a P/T net module s is equal to the number of synchronized firing groups in which the module participates:*

$$\mathrm{cou}(s) = |\{g \in G \mid \exists t \in T_{e_s} : t \in g\}| \tag{42}$$

The coupling of a PTC-system net $\mathcal{SN}$ is equal to the number of its synchronized firing groups:

$$\mathrm{cou}(\mathcal{SN}) = |G| \tag{43}$$

Each synchronized firing group of a PTC-system net corresponds to a transition in its equivalent P/T net connecting places belonging to multiple modules [8]. Therefore, the firing groups directly relate to the inter-module relationships ER. Consequently, it follows that cou satisfies the properties for a coupling measure as outlined in Section 3.5.

As performing the reduction in **Phase 2** leads to the runtime given in Equation (37), it depends on the number of external transitions $|T_e|$ instead of $|G|$. Therefore, another measure of coupling based on the PTC-system net's external transitions arises naturally.

Definition 5. *The coupling of a P/T net module s is equal to the number of its external transitions, and the coupling of a PTC-system net $\mathcal{SN}$ is equal to the number of its external transitions:*

$$\mathrm{cou_e}(s) = |T_{e_s}|, \qquad \mathrm{cou_e}(\mathcal{SN}) = |T_e| \tag{44}$$

The external transitions directly relate to the synchronized firing groups and, therefore, to the inter-module relationships ER. Again, it follows that $\mathrm{cou_e}$ does indeed satisfy the required properties.

Finally, the extent to which entries in δ' can be eliminated in **Phase 2** instead of **Phase 3** captures an important aspect of modularity in PTC-system nets. Indeed, the expression of the remaining columns given in Equation (40) leads to the following measure of cohesion based on the rank of the internal incidence matrix.

Definition 6. *The cohesion of a P/T net module s is equal to the number of internal transitions, whose effect vectors are linearly independent, divided by the number of its places:*

$$\mathrm{coh}(s) = \frac{rank(\lambda_s)}{|P_s|} \tag{45}$$

The cohesion of a PTC-system net $\mathcal{SN}$ is equal to the number of internal transitions, whose effect vectors are linearly independent, divided by the number of all places:

$$\mathrm{coh}(\mathcal{SN}) = \frac{rank(\lambda)}{|P|} \tag{46}$$

In this case, the relation between the rank of the internal incidence matrix and what would be expected of a cohesion measure isn't as direct as for the coupling measures. Therefore, we will show that the expected properties as outlined in Section 3.5 are indeed satisfied.

1. **Non-negativity and Normalization**
   As the rank of a matrix is non-negative and bounded by the minimum between the number of rows and the number of columns, coh is non-negative and normalized:

$$\mathrm{coh}(m) \in [0, 1], \quad \mathrm{coh}(MS) \in [0, 1] \tag{47}$$

2. **Null Value**
   There being no intra-module relationships in a module means that the module doesn't contain any internal transitions. In that case the rank of the internal incidence matrix is equal to 0 for each module and for the entire modular net. Therefore, it follows:

$$IR_m = \emptyset \Rightarrow \mathrm{coh}(m) = 0, \quad IR = \emptyset \Rightarrow \mathrm{coh}(MS) = 0 \tag{48}$$

3. **Monotonicity**
   Intramodule relationships can only be added by adding further internal transitions. If the effect of such transitions is a linear combination of already existing transitions, the rank stays the same. If it isn't a linear combination, the rank increases. Consequently, for a module m' that results from a module m by adding intra-module relationships so that $IR_m \subseteq IR_{m'}$ and the corresponding modular systems MS' (containing m') and MS (containing m), it follows that:

$$\mathrm{coh}(m) \leq \mathrm{coh}(m'), \quad \mathrm{coh}(MS) \leq \mathrm{coh}(MS') \tag{49}$$

4. **Cohesive Modules**
   After merging two unrelated modules, m_1 and m_2, into the combined m', their combined incidence matrix is a block diagonal matrix with the individual incidence matrices as blocks. In this form, it can be directly seen that the rows/columns corresponding to m_1 are necessarily linearly independent of the rows/columns corresponding to m_2. Consequently, the rank of the combined internal incidence matrix is equal to the sum of the ranks of the individual internal incidence matrices:

$$\mathrm{rank}(\lambda_{m'}) = \mathrm{rank}(\lambda_{m_1}) + \mathrm{rank}(\lambda_{m_2}) \tag{50}$$

   For the cohesion follows then:

$$\mathrm{coh}(m') = \frac{\mathrm{rank}(\lambda_{m_1}) + \mathrm{rank}(\lambda_{m_2})}{|P_{m_1}| + |P_{m_2}|} \leq \max\left(\frac{\mathrm{rank}(\lambda_{m_1})}{|P_{m_1}|}, \frac{\mathrm{rank}(\lambda_{m_2})}{|P_{m_2}|}\right) = \max\left(\mathrm{coh}(m_1), \mathrm{coh}(m_2)\right) \tag{51}$$

   Considering the modular system as a whole, both the rank of the internal incidence matrix as well as the number of places stay the same. Therefore, the modular system MS containing m_1 and m_2 has the same cohesion as the modular system MS' in which m_1 and m_2 are replaced by m':

$$\mathrm{coh}(MS) = \mathrm{coh}(MS') \tag{52}$$

   In particular, the desired inequality of $\mathrm{coh}(MS) \geq \mathrm{coh}(MS')$ holds.

Interestingly, this cohesion measure inversely relates to the number of internal P-invariants. By the rank-nullity theorem [23], the dimension of the solution space of a homogenous linear system with matrix A is equal to the difference between the dimension of the domain and the rank of A. In terms of internal P-invariants, this is the difference between the number of places $|P|$ and the rank of the internal incidence matrix $\mathrm{rank}\,\lambda$. It follows then that the cohesion coh is higher if there are fewer internal P-invariants.

9. Transforming Bordered Block Diagonal Matrices into Modular Nets

In Section 8, measures of coupling and cohesion were defined based on properties of matrices that allow more efficient computation of solutions to homogenous linear systems as outlined in Section 7. It follows, then, that this parallel of good modularity between SBBD matrices and modular nets lends itself to considering them as the same kind of optimization problem.

Just as the inherent structure of modular P/T nets translates directly to the structure of the transposed incidence matrix, forming an SBBD matrix [8], the reverse is true as well. If the transposed incidence matrix of a non-modular P/T net is arranged so that it is in SBBD form, then the columns of each block along the diagonal constitute a set of places. Each of these sets is non-empty and disjoint from all other blocks, and all sets together encompass all places. Therefore, they together form a partition of the net's places. This is illustrated by the following example, in which each set of places corresponding to a block along the diagonal constitutes a part in the partition:

$$\Delta^{\mathsf{T}} = \begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 & p_4 \end{matrix} \\ \left(\begin{array}{cc:cc} 1 & -1 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \\ 3 & -1 & -1 & -2 \end{array}\right) \end{array} \quad \longrightarrow \quad \mathcal{P} = \{\{p_1, p_2\}, \{p_3, p_4\}\} \tag{53}$$

Such a partition is sufficient for deducing a modular structure for the initially unmodularized net [34]. It follows then that every procedure for converting an arbitrary matrix to SBBD form directly translates to the automatic modularization of P/T nets.

A matrix structure similar to the SBBD form is the doubly-bordered block diagonal (DBBD) structure, which also allows a border along the rightmost block column of the matrix. In the context of modular P/T nets, this additional border would, in the most direct way, translate to inter-module interactions between places in addition to the interactions between transitions. Therefore, the BDDB structure could be interpreted as a modular net allowing both synchronous channels and shared places.

But the BDDB structure is also relevant in modular P/T nets that use only synchronous channels. Specifically, it is the special case of symmetric matrices in DBBD form that is of interest. When considering the symmetric $\mathbf{pre}^{\mathsf{T}}\mathbf{pre} + \mathbf{post}^{\mathsf{T}}\mathbf{post}$ in DBBD form, its blocks along the diagonal translate to internal transitions, while its borders translate to synchronized firing groups. Therefore, permuting matrices into BDDB form is also analogous to modularization of P/T nets. This is illustrated by the following example:

$$\mathbf{pre}^{\mathsf{T}}\mathbf{pre} + \mathbf{post}^{\mathsf{T}}\mathbf{post} = \begin{array}{c} \begin{matrix} & t_1 & t_2 & t_3 & t_4 & t_5 \end{matrix} \\ \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{matrix} \left(\begin{array}{cc:cc:c} 4 & 2 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 1 & 5 & 3 \\ 0 & 0 & 5 & 2 & 2 \\ \hdashline 2 & 1 & 3 & 2 & 1 \end{array}\right) \end{array} \quad \longrightarrow \quad \begin{aligned} X_{1,2} &= {}^\bullet t_1 \cup t_1^\bullet \cup {}^\bullet t_2 \cup t_2^\bullet \\ X_{3,4} &= {}^\bullet t_3 \cup t_3^\bullet \cup {}^\bullet t_4 \cup t_4^\bullet \\ \mathcal{P} &= \{X_{1,2}, X_{3,4}, P \setminus (X_{1,2} \cup X_{3,4})\} \end{aligned} \tag{54}$$

The transitions corresponding to the blocks in the diagonal would then be internal transitions. In this case, t_1 and t_2 would be internal transitions of the module given by the set of places $X_{1,2}$. The same applies to t_3, t_4 and the module given by $X_{3,4}$. The remaining places $P \setminus (X_{1,2} \cup X_{3,4})$ are only connected to external transitions, which would be t_5. The rank of the blocks in $\mathbf{pre}^{\mathsf{T}}\mathbf{pre} + \mathbf{post}^{\mathsf{T}}\mathbf{post}$ then relates back to the rank of the blocks in Δ^{T} [35, 36].

Permuting matrices into SBBD or DBBD form is a well-researched topic [25, 37, 38, 39, 24]. The common goal of these transformations of minimizing the border size while keeping the size of the diagonal blocks balanced [38] also optimizes the modular structure of the corresponding net. The underlying problem, formulated as the partitioning of hypergraphs, was already examined in the context of P/T net modularization in [34]. However, the extensive literature could inform more specific approaches to this problem, enabling more efficient modularization of P/T nets.

10. Discussion

The analysis of the modular Gaussian elimination naturally leads to notions of coupling and cohesion. They allow coupling and cohesion in modular P/T nets to be formally defined. Further, by being based on properties that directly influence the runtime of computing P-invariants, loose coupling and high cohesion have been formally demonstrated to be desirable properties of modular P/T nets. Therefore, we believe it is justified to use these measures to judge the quality of modularity. They could be used

to inform decisions in designing modular systems or serve in determining the quality of outputs of automatic modularization procedures.

It is also conceivable to use them in modularization procedures directly. For example, a probabilistic algorithm could output many different possible ways of splitting a net into modules. It would then be required to determine which of these potential modularizations is in some sense the best, which could be done through our coupling and cohesion measures.

Another interesting aspect of the reduction in **Phase 1** is that having multiple modules that only partially exhibit the same internal behavior is already beneficial. This would justify using the modularization based on replication proposed in [34] to also find partial replications.

One aspect that wasn't further examined in this paper is that the structure of the synchronized incidence matrix δ could be further utilized. In practice, not every module will be participating in every synchronized firing group, so that many entries in δ could be zeros. However, instead of specifically addressing this through modifications in the modular Gaussian elimination, it might be more prudent to instead arrange the matrix in a recursive bordered block diagonal form [40, 37]. To describe the corresponding modular net, a recursive definition would be appropriate, in which not only P/T nets but also PTC-system nets themselves would act as potential modules.

Lastly, it is worth mentioning that the use of the modular Gaussian elimination isn't solely confined to the computation of P-invariants. For example, just the step of reducing an incidence matrix to row echelon form is enough to identify redundant information, reducing memory resources required for state space construction [41].

11. Conclusion

An algorithm for performing Gaussian elimination in a modular manner has been presented and analyzed. It aims to exploit the independence between modules resulting from the structure of modular nets as much as possible. Its analysis focuses on how the specifics of this structure influence the algorithm's runtime.

The parts of the algorithm that benefit from the independence (**Phase 1** and **Phase 2**) achieve asymptotic improvements if the number of modules depends on the net's overall number of places and transitions as opposed to having a fixed number of modules regardless of the net's overall size. This is in particular the case if modules are of roughly the same constant size.

The size of the interface between modules influences the algorithm's runtime of parts, in which the terms corresponding to interactions between modules are considered (**Phase 2** and **Phase 3**). The specific ways of measuring this interface size, either through the external transitions in the modules or the resulting possible synchronization, naturally serve as formal measures of coupling in modular P/T nets.

The extent to which the computation concerning interactions between modules can be performed in a manner that keeps modules independent is determined by the rank of the internal incidence matrix. This rank, then, constitutes a formal measure of cohesion in modular P/T nets.

These measures of coupling and cohesion are thus based on characteristics of modular nets that are demonstrably beneficial in the verification of system properties. Due to parallels in the good modularity in this sense for both matrices and modular nets, it naturally arises that procedures for transforming matrices into modular structures and the modularization of P/T nets aim to optimize the same goal. Specifically, procedures for permuting matrices into either SBBD or DBBD form can be directly used to transform a P/T net into a modular P/T net.

Further investigations, as discussed in the previous section, are quite promising. The concept of modules is highly relevant to the design of system models and analysis. The role of the concept of nets-within-nets is central for the design of complex interacting systems. Therefore, the topic of coupling and cohesion will be one of our research interests.

Declaration on Generative AI

During the preparation of this work, the authors used …

- …LanguageTool, QuillBot in order to: Grammar and spelling check
- …DeepL in order to: Translate Text.
- …ChatGPT in order to: Rephrasing.
- …Grammarly in order to: Grammar and spelling check, Repharsing.

After using these tool(s)/service(s), the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. L. Parnas, On the criteria to be used in decomposing systems into modules, Communications of the ACM 15 (1972) 1053–1058.

[2] P. Fettke, W. Reisig, Understanding the Digital World – Modeling with HERAKLIT, Springer, 2024. URL: https://doi.org/10.1007/978-3-031-61898-7. doi:10.1007/978-3-031-61898-7.

[3] 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.

[4] R. Valk, Petri nets as dynamical objects, in: G. Agha, F. D. Cindio (Eds.), 16th Intern. Conf. on Application and Theory of Petri Nets, Torino, Italy, Workshop proceedings, University of Torino, 1995. Workshop während der "'16th International Conference on Application and Theory of Petri Nets"', Turin, Italien, 26.–30. Juni, 1995.

[5] R. Valk, Bridging the gap between place- and Floyd-invariants with applications to preemptive scheduling, in: A. Marsan, M. (Ed.), Application and Theory of Petri Nets 1993, Proceedings 14th International Conference, Chicago, Illinois, USA, volume 691 of *Lecture Notes in Computer Science*, Springer-Verlag, 1993, pp. 433–452.

[6] 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.

[7] 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.

[8] S. Bott, D. Moldt, L. Clasen, M. Hansson, Invariant calculations for P/T-nets with synchronous channels, in: M. Köhler-Bußmeier, D. Moldt, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2024 co-located with the 45th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2024), June 24 - 25, 2024, Geneva, Switzerland, volume 3730 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 104–121. URL: https://ceur-ws.org/Vol-3730.

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

[10] 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.

[11] 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.

[12] S. Bott, Untersuchung algorithmischer Optionen für die effiziente Invariantenberechnung in P/T-Netzen mit synchronen Kanälen und der technischen Umsetzung in Renew, Bachelor thesis, Vogt-Kölln Str. 30, D-22527 Hamburg, 2024.

[13] L. Petrucci, Cover picture story: Experiments with modular state spaces, Petri Net Newsletter 68 (2005) cover–and.

[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] L. C. Briand, S. Morasca, V. R. Basili, Property-based software engineering measurement, IEEE transactions on software Engineering 22 (2002) 68–86.

[16] E. B. Allen, T. M. Khoshgoftaar, Measuring coupling and cohesion: An information-theory approach, in: Proceedings sixth international software metrics symposium (cat. no. pr00403), IEEE, 1999, pp. 119–127.

[17] E. B. Allen, Measuring graph abstractions of software: an information-theory approach, in: Proceedings eighth ieee symposium on software metrics, IEEE, 2002, pp. 182–193.

[18] D. Poshyvanyk, A. Marcus, The conceptual coupling metrics for object-oriented systems, in: 2006 22nd IEEE International Conference on Software Maintenance, IEEE, 2006, pp. 469–478.

[19] S. Kramer, H. Kaindl, Coupling and cohesion metrics for knowledge-based systems using frames and rules, ACM Transactions on Software Engineering and Methodology (TOSEM) 13 (2004) 332–358.

[20] S. Moser, V. B. Misic, Measuring class coupling and cohesion: a formal metamodel approach, in: Proceedings of Joint 4th International Computer Science Conference and 4th Asia Pacific Software Engineering Conference, IEEE, 1997, pp. 31–40.

[21] S. Morasca, Measuring attributes of concurrent software specifications in petrinets, in: Proceedings Sixth International Software Metrics Symposium (Cat. No. PR00403), IEEE, 1999, pp. 100–110. doi:10.1109/METRIC.1999.809731.

[22] R. K. George, A. Ajayakumar, A Course in Linear Algebra, 1 ed., Springer Nature Singapore, 2024. doi:10.1007/978-981-99-8680-4.

[23] J. Liesen, V. Mehrmann, Linear algebra, Springer, 2015.

[24] I. S. Duff, J. A. Scott, Stabilized bordered block diagonal forms for parallel sparse solvers, Parallel Computing 31 (2005) 275–289.

[25] O. Kaya, E. Kayaaslan, B. Uçar, I. S. Duff, Fill-in reduction in sparse matrix factorizations using hypergraphs, Research Report RR-8448, INRIA, 2014. URL: https://inria.hal.science/hal-00932882.

[26] P. Deuflhard, A. Hohmann, Numerische Mathematik: Bd. 1: Eine algorithmisch orientierte Einführung, Walter de Gruyter, 2008.

[27] 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.

[28] S. Boyd, L. Vandenberghe, Introduction to applied linear algebra: vectors, matrices, and least squares, Cambridge university press, 2018.

[29] V. Strassen, Gaussian elimination is not optimal, Numerische mathematik 13 (1969) 354–356.

[30] J. R. Bunch, J. E. Hopcroft, Triangular factorization and inversion by fast matrix multiplication, Mathematics of Computation 28 (1974) 231–236.

[31] F. L. Gall, F. Urrutia, Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2018, pp. 1029–1046.

[32] C. Camarero, Simple, fast and practicable algorithms for cholesky, lu and qr decomposition using fast rectangular matrix multiplication, arXiv preprint arXiv:1812.02056 (2018).

[33] S. U. A. (https://math.stackexchange.com/users/1154/), How do you show monotonicity of the ℓ^p

norms?, Mathematics Stack Exchange, 2010. URL: https://math.stackexchange.com/q/4122, visited on 2024-03-27.

[34] J. Gaede, J.-H. Overath, S. Wallner, Automatic modularization of place/transition nets, in: M. Köhler-Bußmeier, D. Moldt, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2024 co-located with the 45th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2024), June 24 - 25, 2024, Geneva, Switzerland, volume 3730 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 53–73. URL: https://ceur-ws.org/Vol-3730.

[35] G. Matsaglia, G. PH Styan, Equalities and inequalities for ranks of matrices, Linear and multilinear Algebra 2 (1974) 269–292.

[36] G. Marsaglia, G. P. Styan, When does rank (a+ b)= rank (a)+ rank (b)?, Canadian Mathematical Bulletin 15 (1972) 451–452.

[37] Ü. V. Çatalyürek, C. Aykanat, E. Kayaaslan, Hypergraph partitioning-based fill-reducing ordering for symmetric matrices, SIAM Journal on Scientific Computing 33 (2011) 1996–2023.

[38] C. Aykanat, A. Pinar, Ü. V. Çatalyürek, Permuting sparse rectangular matrices into block-diagonal form, SIAM Journal on scientific computing 25 (2004) 1860–1879.

[39] Z. Kukelova, M. Bujnak, J. Heller, T. Pajdla, Singly-bordered block-diagonal form for minimal problem solvers, in: Asian Conference on Computer Vision, Springer, 2014, pp. 488–502.

[40] B. Fagginger Auer, R. Bisseling, A geometric approach to matrix ordering, arXiv e-prints (2011) arXiv–1105.

[41] 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.